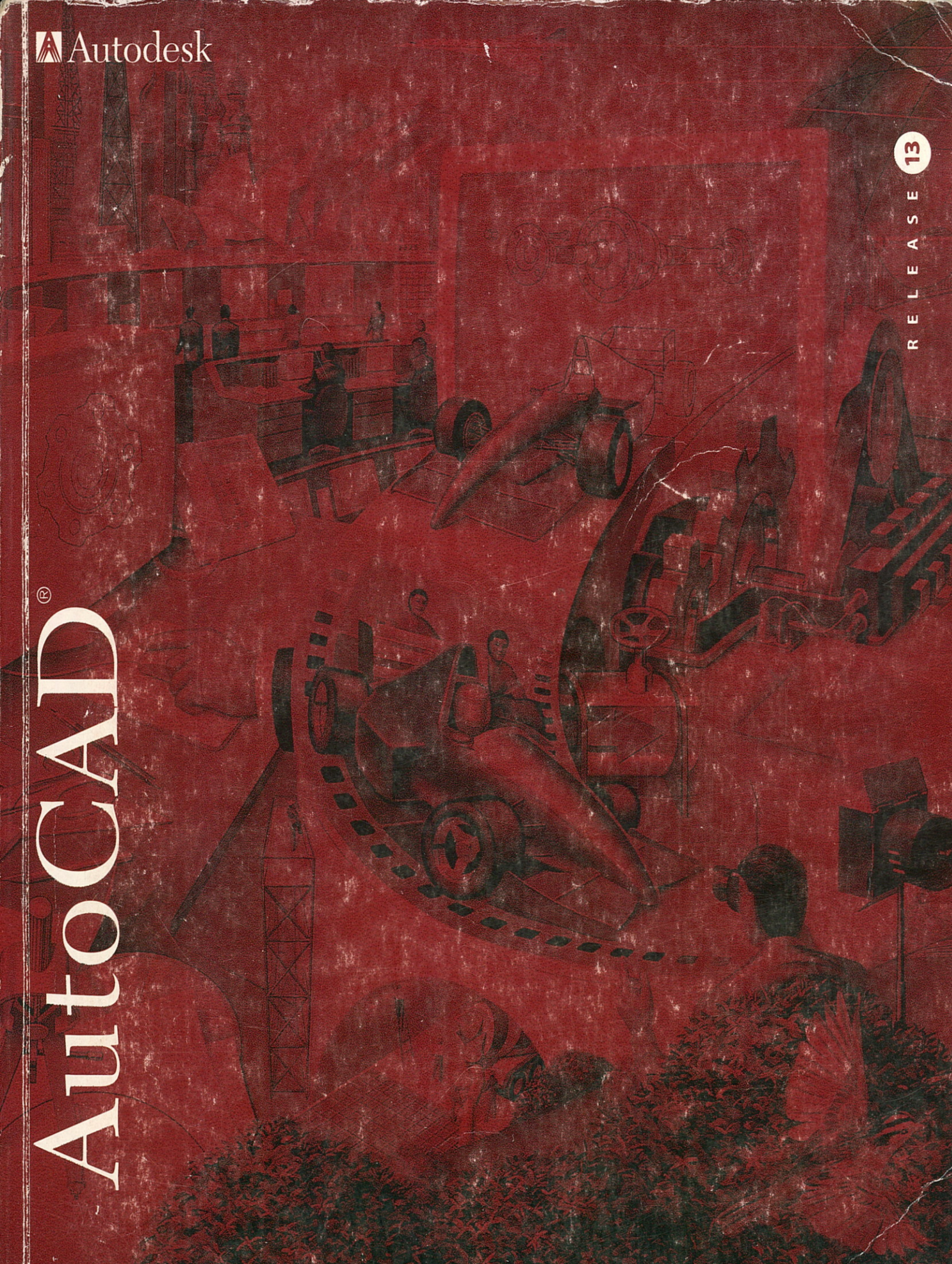
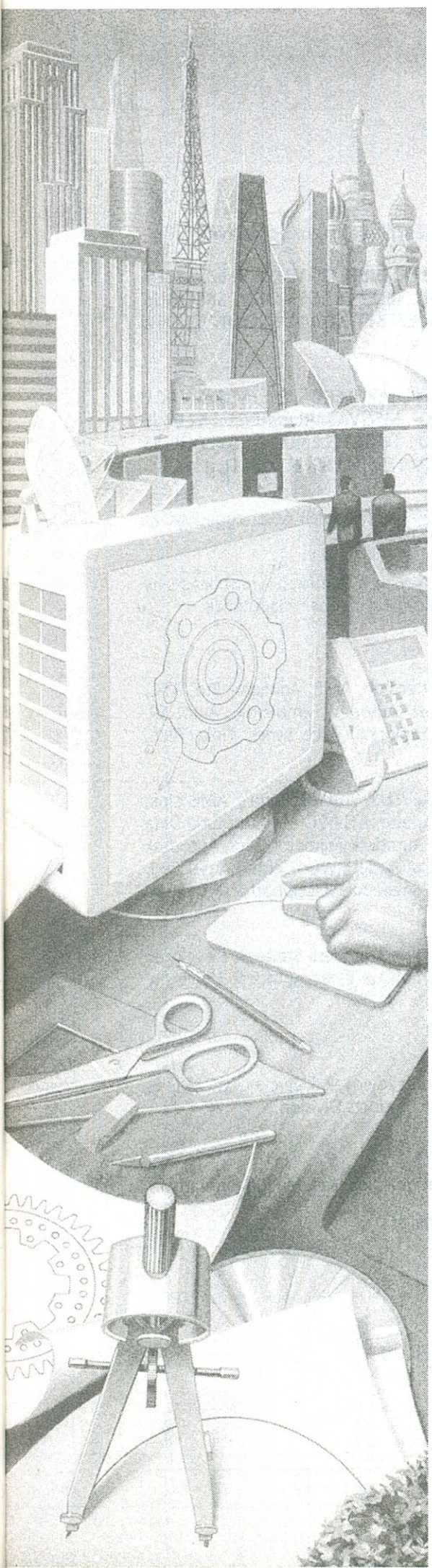
Autodesk

AutoCAD®

RELEASE 13



ALKALMAZÁSHOZ IGAZÍTÁSI ÚTMUTATÓ



Autodesk®

AutoCAD® Release 13

ALKALMAZÁSHOZ IGAZÍTÁSI ÚTMUTATÓ

00105-140000-5040

Copyright © 1994 Autodesk, Inc.

Minden jog fenntartva

A jelen kiadvány, illetve annak bármilyen része, semmilyen formában, módszerrel és céllal nem sokszorosítható.

AZ AUTODESK, INC. A JELEN ANYAGRA VONATKOZÓAN SEMMILYEN KÖZVETETT VAGY KÖZVETLEN FELELŐSSÉGET NEM VÁLLAL, BELEÉRTVE A KERESKEDELMI FORGALMAZÁSBÓL, ILLETVE EGY ADOTT SPECIÁLIS CÉLRA TÖRTÉNŐ FELHASZNÁLÁSBÓL ADÓDÓ KÖZVETETT FELELŐSSÉGET IS. EZEKET AZ ANYAGOKAT CSAK A JELENLEG MEGTEKINTETT FORMÁBAN BOCSÁJTJA RENDELKEZÉSRE.

AZ AUTODESK, INC. SEMMILYEN ESETBEN NEM TEHETŐ FELELŐSSÉ SPECIÁLIS, JÁRULÉKOS, ESETLEGES VAGY KÖVETKEZMÉNYKÉNT EREDŐ KÁROKÉRT, AMELYEK A JELEN ANYAG MEGVÁSÁRLÁSÁBÓL ILLETVE HASZNÁLATÁBÓL ADÓDNAK. AZ AUTODESK, INC. CÉGET - AZ ELJÁRÁSTÓL FÜGGETLENÜL - EGYEDÜL ÉS KIZÁRÓLAGOSAN CSAK AZ ITT ISMERTETETT ANYAGOK VÁSÁRLÁSI ÉRTÉKÉIG TERHELI FELELŐSSÉG.

A jelen anyagoknak az angoltól eltérő nyelvű publikációkban történő felhasználásának feltételeit és az engedélyt az Autodesk, Inc. adhatja meg.

Az Autodesk, Inc. fenntartja magának a jogot, hogy belátása szerint változtassa és javítsa a terméket. A jelen kiadvány a termék nyomtatáskori állapotát tükrözi, és nem feltétlenül felel meg a termék bármikori jövőbeli állapotának.

Autodesk védjegyek

Az alábbiak az Autodesk, Inc. bejegyzett védjegyei: ADI, Advanced Modeling Extension, AME, Animator Pro, ATC, AutoCAD, AutoCAD Development System, Autodesk, Autodesk Animator, az Autodesk embléma, AutoLISP, AutoShade, AutoSketch, AutoSolid, AutoSurf, Geodyssey, HOOPS, Multimedia Explorer, Office Series, TinkerTech, World-Creating Toolkit, 3D Plan és a 3D Studio.

Az alábbiak az Autodesk, Inc. védjegyei: ACAD, Advanced User Interface, AEMULUS, AEMULUS(mf), AME Link, Animation Partner, Animation Player, Animation Pro Player, A Studio in Every Computer, ATLAST, AUI, AutoCAD Data Extension, AutoCAD Simulator, AutoCAD SQL Extension, AutoCAD SQL Interface, Autodesk Animator Clips, Autodesk Animator Theatre, Autodesk Device Interface, Autodesk Software Developer's Kit, Autodesk WorkCenter, AutoCDM, AutoEDM, AutoFlix, AutoLathe, AutoVision, DXF, FLI, Flic, Generic 3D, SketchTools, SmartCursor, Supportdesk, Texture Universe, Transforms Ideas Into Reality és a Visual Link.

Az alábbiak az Autodesk, Inc. szolgáltatás-védjegyei: Autodesk Strategic Developer, az Autodesk Strategic Developer embléma, Autodesk Registered Developer, az Autodesk Registered Developer embléma és az Autodesk University embléma.

A Generic Software, Inc. védjegyei

A Generic, a Generic CADD, a Generic Software és a Generic 3D Drafting bejegyzett védjegyek, illetve a GenCADD és a Home Series védjegyek tulajdonosa a Generic Software, Inc., dba Autodesk Retail Products Division.

Külső fejlesztők védjegyei

International CorrectSpell™ spelling correction system © 1993 by InfoSoft International, Inc. Minden jog fenntartva.

A RenderMan a Pixar bejegyzett védjegye. Az Autodesk a Pixar engedélyével használja.

Az összes többi védjegy mindenkori tulajdonosának védjegye.

TARTALOM

BEVEZETÉS 1

I. RÉSZ ALKALMAZÁSHOZ IGAZÍTÁSI REFERENCIA 15

1. FEJEZET ■ AZ ADAPTÁLÁS ALAPJAI 17

2. FEJEZET ■ VONALTÍPUSOK ÉS SRAFFOZÁSI MINTÁK 37

3. FEJEZET ■ ALAKOK, FONTOK ÉS A POSTSCRIPT TÁMOGATÁSA 49

4. FEJEZET ■ SAJÁT MENÜK 85

5. FEJEZET ■ AZ ÁLLAPOTSOR KONFIGURÁLÁSA ÉS A DIESEL KARAKTERES KIFEJEZÉSNYELV 139

6. FEJEZET ■ PROGRAMOZÁSI FELÜLETEK 157

II. RÉSZ AUTOLISP REFERENCIA 187

7. FEJEZET ■ AZ AUTOLISP ALAPJAI 189

8. FEJEZET ■ ÁLTALÁNOS SEGÉDFÜGGVÉNYEK 223

9. FEJEZET ■ KIVÁLASZTÁSI HALMAZOKAT, OBJEKTUMOKAT ÉS SZIMBÓLUMTÁBLÁKAT KEZELŐ FÜGGVÉNYEK 263

10. FEJEZET ■ PÁRBESZÉDPANELEK KEZELÉSE 301

11. FEJEZET ■ AUTOLISP GYAKORLÓ PÉLDA 327

12. FEJEZET ■ AZ AUTOLISP FÜGGVÉNYEK ÁTTEKINTÉSE 345

13. FEJEZET ■ AUTOLISP FÜGGVÉNYEK KATALÓGUSA 373

14. FEJEZET ■ A KÜLSŐLEG DEFINIÁLT PARANCSOK ÉS RENDSZERVÁLTOZÓK HASZNÁLATA 529

15. FEJEZET ■ MEMÓRIAKEZELÉS 563

16. FEJEZET ■ ALKALMAZÁS-SPECIFIKUS DXF CSOPORTKÓDOK 571

17. FEJEZET ■ AUTOLISP HIBAKÓDOK ÉS HIBAÜZENETEK 597

III. RÉSZ PROGRAMOZHATÓ PÁRBESZÉDPANEL REFERENCIA 615

18. FEJEZET ■ PROGRAMOZHATÓ PÁRBESZÉDPANELEK 617

19. FEJEZET ■ A PÁRBESZÉDVEZÉRLŐ NYELV 649

FÜGGELÉKEK 685

A. FÜGGELÉK ■ ASCII KÓDOK 689

B. FÜGGELÉK ■ FÁJLOK LISTÁJA 693

TÁRGYMUTATÓ 701

BEVEZETÉS

Az AutoCAD olyan nyílt felépítésű, általános célú tervezőrendszer, melynek számos tulajdonságát kiterjesztheti és saját igényei szerint alakíthatja. Ennek eredményeképpen az AutoCAD programot szükség szerint bővítheti és formálhatja. Az AutoCAD Adaptációs kézikönyvében találhatók az ilyen és más testreszabási (adaptációs) jellemzők létrehozásához szükséges részletes tudnivalók és eljárások.

Az AutoCAD program a következő adaptációk létrehozására nyújt lehetőséget:

- DOS programokat és segédprogramokat futtathat AutoCAD környezetből
- A gyakran használt parancsokat rövidített álnevekkel láthatja el
- On-line súgófájlokat hozhat létre saját alkalmazásaihoz és eljárásaihoz
- Saját menüket hozhat létre
- Az ismétlődő feladatokat automatikussá teheti
- Automatizálhatja az ismétlődő feladatok megoldásának végrehajtását
- AutoCAD parancsokat definiálhat át vagy iktathat ki

- Létrehozhatja saját alkalmazásait és párbeszédpaneljeit

Terjesztőjénél további olyan függetlenül kifejlesztett alkalmazásokhoz és programbővítésekhez juthat hozzá, amelyekkel még jobban igényeihez igazíthatja az AutoCAD programcsomagot.

Az AutoCAD documentáció használata

Az AutoCAD *Alkalmazáshoz igazítási útmutatója* tapasztalt felhasználóknak készült. Az AutoCAD jellemzőinek és eljárásainak részletes leírását az *on-line súgóban és a Felhasználói útmutatóban* találja meg. Az AutoCAD Release 13 verziójának teljes leírása az AutoCAD CD-ROM lemezéről érhető el. A leírásról a *Dokumentációs útmutatóban* talál további információkat.

Megjegyzés A legfrissebb AutoCAD-információk és eljárások a readme.doc fájlban találhatók.

Az Alkalmazáshoz igazítási útmutató használata

Az AutoCAD adaptációs lehetőségeinek könnyebb megértéséhez az *Alkalmazáshoz igazítási útmutatót* három részre osztottuk:

Első rész: Az *Adaptációs referencia* az AutoCAD általános adaptációs jellemzőit adja meg.

Második rész: Az *AutoLISP referencia* leírja az AutoCAD programhoz illesztett AutoLISP programozási felületet. Az AutoLISP programozási nyelv segítségével az AutoCAD programon belül futtatható saját alkalmazásokat és parancsokat hozhat létre.

Harmadik rész: A Programozható párbeszédpanel referenciája ismerteti, hogyan programozzuk a felhasználó által definiált alkalmazásokhoz kapcsolódó párbeszédpaneleket.

Az *Alkalmazáshoz igazítási útmutatót* úgy állították össze, hogy az AutoCAD Release 13 összes változatával használható legyen. Az AutoCAD *Üzembehelyezési útmutatójában* meggyőződhet arról, hogy az Ön által használt AutoCAD működése mennyiben különbözik a könyvben leírtaktól.

Tipográfiai jelölések

Az AutoCAD képernyőjén megjelenő jellemzők közötti eligazodás megkönnyítésére a speciális szakszavakat az alapszövegtől eltérő betűképpel szedtük. Ennek megfelelően az AutoCAD dokumentációjában a következő jelölések használatosak.

Alaki jelölések

Szövegrész	*Példa*
Lenyomott billentyűk A ↵ billentyű megfelel az Enter és a Return billentyűknek	Delete, Esc
Egyszerre lenyomott billentyűk	Ctrl+C
A menü elemei, a promptok, és a képernyőn megjelenő egyéb szöveg sans serif betűtípusban	A Fájl menüből válassza a Megnyit parancsot! Parancs: **terület** <Első pont>/Objektum/Összead/Kivon:
A begépelendő szöveg félkövér sans serif betűtípusban	Írja be a parancssorba: **alak**
Fájl- és könyvtárnevek, promptsor utáni megjegyzések dőlt betűvel	Válasszon objektumokat: *Használjon egy objektumkiválasztó módszert* *disc.dwg* *c:\acad\support*
Az AutoCAD által elnevezett objektumok, mint például fóliák, vonaltípusok, stílusok, AutoCAD parancsok, rendszerváltozók nagybetűvel	MENTÉS, MÁSOLÁS, DWGNÉV, BLIPMÓD DISC-FRONT, DASHDOT, STANDARD
Az AutoLISP és a C programnyelvek változói, mintaprogramok és ASCII fájlbeli szövegek monospaced stílusban	`***POP1` A pi változóhoz pi értékét rendeljük hozzá
Az AutoLISP, az ADS, és a DIESEL függvénynevek félkövér monospaced stílusban	**`command ads_command( )`**
Függvénydefiníciók alaki változói dőlt monospaced stílusban	A *`string`* és a *`mode`* függvényparaméterek

Változások és fejlesztések a Release 13 programban

Az AutoCAD Release 13 verziójában az AutoCAD számos adaptációs tulajdonsága kibővült, vagy megváltozott. Az *Alkalmazáshoz igazítási referenciában (első rész)* a következő változások ismertetése található:

- Megváltozott az AutoCAD súgófájl formátuma
- A vonaltípusok kiegészültek; így beágyazott alakokat és szövegobjektumokat is elfogadnak
- A Unicode fontdefiníciók gondoskodnak a karakter-kódlapok közötti pontos karakterleképezésről
- Az AutoCAD for Windows menüfájljai számos új tulajdonsággal egészültek ki

A *AutoLISP Referenciában (második rész)*, található fejlesztések és változások a következők:

- Az objektumazonosító-sorszámok állandóan be vannak kapcsolva
- A szimbólumtáblák bejegyzései kapcsolódnak a rajzelemek nevéhez és a grafikus rajzelemekhez hasonlóan kezelhetőek.
- Egy új, nem grafikus DICTIONARY rajzelem (hasonló a szimbólumtáblához) gondoskodik a többvonalas stílusok és csoportok tárolásáról.
- Új DXF csoportkódokat hoztak létre az új AutoCAD rajzelemek támogatására.
- Az új AutoLISP függvények a következők:

  arx megadja az aktuális betöltött ARX alkalmazások listáját

  arxload betölt egy ARX alkalmazást

  arxunload eltávolít egy ARX alkalmazást

  ase_docurrent tájékoztat a jelenlegi ASE beállításokról

  ase_dolist visszakeresi az adatbázis-objektumok elérési útvonalának listáját

  ase_dopathcode visszakeresi az adatbázis-objektumok definíciós kódját a kijelölt elérési útvonalra

  ase_dopathmake alapegységek neveiből létrehozza az adatbázis-objektum elérési útvonalát

  ase_dopathname az adatbázis-objektum elérési útvonalából megadja a kiválasztott nevet

  ase_dostatus megadja a kiválasztott objektum állapotát

ase_errcode visszakeresi az ASE hibakódot

ase_errdsc visszakeresi az ASE hibát jellemző kódot

ase_errmsg az ASE hiba leírásával tér vissza

ase_errqty megadja a hibaleírók számát

ase_linkdelete eltávolítja a csatolás-azonosító által kijelölt csatolást

ase_linkget visszakeresi a csatolás-azonosító által kijelölt csatolás adatait

ase_linkmake létrehoz egy új csatolást

ase_linkupdate aktualizálja a már létező csatolást

ase_lpcurrent megadja az aktuális csatolás elérési útvonalát

ase_lperase törli a csatolás elérési útvonal neve által kijelölt csatolás- elérési útvonalat

ase_lpisupdatable ellenőrzi, hogy aktualizálható-e a teljes csatolás elérési útvonala

ase_lpkey visszakeresi a megadott csatolási tábla kulcsoszlop neveit

ase_lplist visszakeresi a csatolás elérési útvonalneveinek listáját

ase_lpnew új csatolási útvonalat hoz létre

ase_lppath visszaadja a megadott csatolás elérési útvonal nevéhez tartozó tábla elérési útvonalát

ase_lprename átnevezi a csatolás elérési útját

ase_lsadd a csatolás-azonosítót a csatolás kiválasztáshoz rendeli

ase_lscopy új helyre másolja a csatolás kiválasztást

ase_lscreate a megadott csatolószűrő szerint létrehoz egy csatolás kiválasztást

ase_lsdel a csatolás kiválasztásból törli a csatolási pozicíót

ase_lsentsel visszakeresi azokat a kiválasztott objektumokat (rajzelemeket), amelyek a csatolás kiválasztásban meglevő csatoláshoz kapcsolódnak

ase_lserase törli a kapcsolódásokat, és megszünteti a csatolás kiválasztást

ase_lsfree felszabadítja a csatolás kiválasztást

ase_lsget megadja a csatolás kiválasztás meghatározott helyéhez tartozó csatolás-azonosítót

ase_lsintersect a megadottak kivételével törli az összes csatolás-azonosítót a csatolás kiválasztásból

ase_lsmemb ellenőrzi, hogy a csatolás-azonosító a csatolás kiválasztáshoz tartozik-e

ase_lsqty megadja a csatolás kiválasztásban a csatolások számát

ase_lssubtract törli a megadott csatolás-azonosítókat a csatolás kiválasztásból

ase_lsunion egyesít két csatolás kiválasztást

ase_lsxnames visszaadja a csatolás kiválasztásban található egymással összefüggő xref vagy blokknevek listáját

ase_version megadja az aktuális ASE verziószámot

autoarxload parancsneveket definiál előre egy kiválasztott ARX fájl betöltéséhez

autoload parancsneveket definiál előre egy kiválasztott AutoLISP fájl betöltéséhez

autoxload előre definiálja a parancsneveket egy kapcsolódó ADS alkalmazáshoz

dictnext megtalálja a következő tételt egy könyvtárban

dictsearch megkeres egy tételt egy könyvtárban

getcfg visszakeresi az alkalmazás adatait az *acad.cfg* fájl AppData részéből

help bármilyen környezetből meghívható súgó lehetőség

namedobjdict megadja az aktuális rajzban azon objektumkönyvtár nevét, amely a rajz összes, nem grafikus objektumának gyökérkönyvtára

setcfg az *acad.cfg* fájl AppData részébe írja az alkalmazás adatait

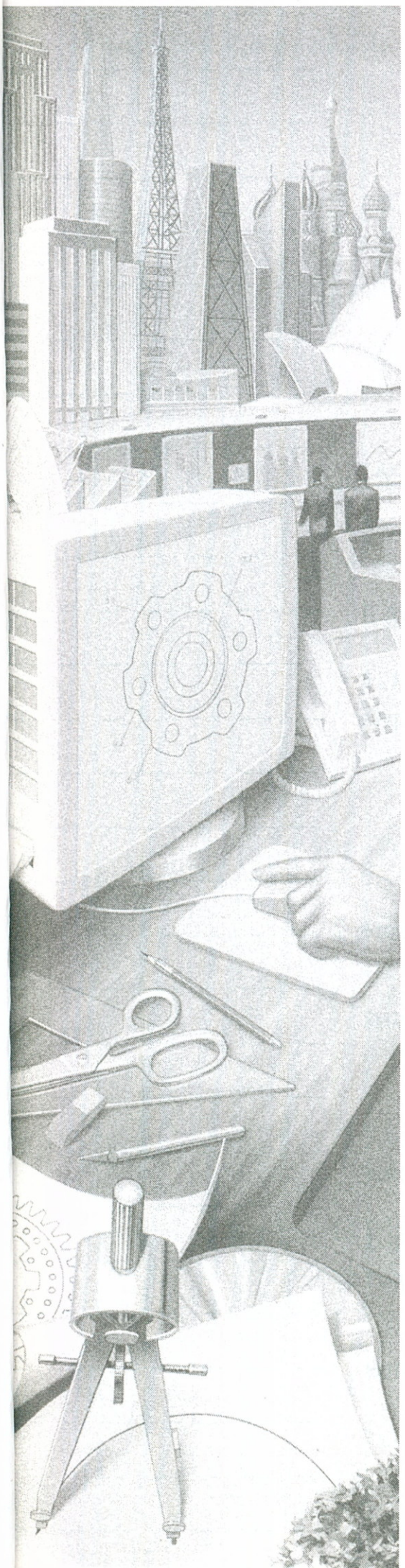
setfunhelp rögzít egy a felhasználó által definiált parancsot a Súgóban; így szükség esetén a megfelelő Súgófájl és témakör jelenik meg

snvalid ellenőrzi a karakterek érvényességét a szimbólumtábla nevében

startapp elindít egy Windows alkalmazást

tblobjname megadja egy kiválasztott szimbólumtábla bejegyzés nevét

A *Programozható párbeszédpanel referenciában (harmadik rész)* eszközölt egyetlen változás, hogy a mnemonik jeleket a (&) szimbólum segítségével jelölhetjük meg egy címke karakterláncában.



I. RÉSZ

ALKALMAZÁSHOZ IGAZÍTÁSI REFERENCIA

TARTALOM

I. RÉSZ ALKALMAZÁSHOZ IGAZÍTÁSI REFERENCIA 15

1. FEJEZET ■ AZ ADAPTÁLÁS ALAPJAI 17

Alapkövetelmények 18
Az AutoCAD környezete 18
A könyvtárkeresési útvonal 18
Kiegészítő fájlok elérési útvonala 19
Egy példa könyvtár felépítésére 19
Parancskeresési eljárás 22
Módosítható kiegészítő fájlok 23
Programparaméter-fájl—*acad.pgp* 25
Külső parancsok definiálása 26
Parancsálnevek 28
Saját súgófájlok 29
Az AutoCAD súgófájl formátuma 30
Speciális formázás 32
Hiperszöveg-csatolások 33
R12 súgófájlok konvertálása R13 súgófájlokká 34
AutoCAD üzenetfájl 36

2. FEJEZET ■ VONALTÍPUSOK ÉS SRAFFOZÁSI MINTÁK 37

Vonaltípus-definíciós fájlok 38
Egyszerű vonaltípusok 39
Összetett vonaltípusok 40
Sraffozási minták készítése 44
Az AutoCAD mintafájlja—*acad.pat* 44
A sraffozott minták felépítése 45

Minták szaggatott vonalakból 45

3. FEJEZET ■ ALAKOK, FONTOK ÉS A POSTSCRIPT TÁMOGATÁSA 49

Az alak- és fontfájlok fordítása 50
PostScript Fontok fordítása 50
Hibák a Postscript fontokban 51
Figyelmeztetések és korlátozások 51
Alakdefiniáló fájlok 51
Alakleírás 51
A vektorhossz- és iránykódok 53
Speciális kódok 54
Szövegfontok leírása 61
Speciális szövegfontok leírása 63
A big-font definiálása 63
Bővített big-fontfájl definíciók 63
A big-font használata 70
Unicode fontleírások 71
Fájlformátum 72
A Unicode fontok használata 72
Fejlett alakdefiniáló módszer 74
PostScript támogatás 75
PostScript képek átadása 75
Az AutoCAD PostScript kiegészítő fájl—*acad.psf* 76
Fejlett PostScript feldolgozások 81
PostScript képek beolvasása 82
Az AutoCAD PostScript értelmező—*acadps* 82
Az AutoCAD fonttábla-fájl—*fontmap.ps* 83
Hibák a beolvasott PostScript képekben 84
PostScript referenciák 84

4. FEJEZET ■ SAJÁT MENÜK 85

Menüfájlok 86
Menüfájl-típusok 86
Menüfájlok betöltése 87
Menük betöltése DOS és UNIX környezetben 88
Menük betöltése Windows alatt 89
Alkalmazási és képernyőmeghajtó menük 89
A menüfájl felépítése 90
Menüszakaszok 90
Almenük 91
Menüpontok megadási módja 94
Menüpontok címkéi 95
A MENUCTL rendszerváltozó 95

Menüpont lezárása 95
Adatbeviteli szünet 97
Idegen nyelvek támogatása a menükben 98
Parancs megszakítása 98
Az echo és a promptok 98
Vezérlőkarakterek a menüpontokban 99
Hosszú menüpontok 100
Menüpont ismétlése 100
Az egyedi objektumkiválasztási módszer használata 101
Feltételes makrókifejezések 101
A menük megadási módjának áttekintése 102
Nyomógomb- és kiegészítő menük 103
A visszahajló törtjel speciális alkalmazása 105
Képernyőmenük 106
Almenücímkék 107
Menüpont-címkék 109
Legördülő és kurzormenük 110
Legördülő és kurzormenük megadási módja 110
Címek a menüsorban 113
Lépcsőzetes almenük 114
Menüpontok tagolása 115
Menüpontok megjelenítésének szabályozása 115
Címkék letiltása 115
Címkék megjelölése 116
Egyidejű letiltás és megjelölés 117
Hivatkozás egy legördülő vagy kurzor-almenüre 117
Hozzáférés a címkék státuszához AutoLISP és ADS függvényekkel 118
Legördülő menük cseréje 119
Megszorítások 119
Ábrazóna-menük 119
Az ábrazóna-menü megadási módja 120
Menüpont-címkék 120
Az ábrazóna-menü megjelenítése 121
Egy tétel kiválasztása az ábrazóna-menüből 121
Minták az ábrazóna-menüre 121
Az ábrazóna-menük diáinak elkészítése 123
Táblamenük 124
Az AutoLISP használata menükben 126
Windows-specifikus menüjellemzők 127
Névcímkék és menücsoportok 128
Részleges menük 129
Részleges menük betöltése 129
Részleges menük törlése 130
Eszköztárak 131

Menü-specifikus súgó 134
Menüpontok súgója 134
Állapotsor-súgó—Súgó karakterláncok 134
Gyorsító billentyűk 135

5. FEJEZET ■ AZ ÁLLAPOTSOR KONFIGURÁLÁSA ÉS A DIESEL KARAKTERES KIFEJEZÉSNYELV 139

Az állapotsor konfigurálása—MODEMACRO 140
A MODEMACRO változó 140
MODEMACRO definíciók 140
MODEMACRO definíciók az AutoLISP használatával 142
DIESEL kifejezések a menükben 144
DIESEL kifejezések az AutoLISP rutinokban 148
Hibakeresés a DIESEL kifejezésekben—MACROTRACE 148
DIESEL karakterlánc-függvények katalógusa 149
+ (összeadás) 149
– (kivonás) 149
*(szorzás) 149
/ (osztás) 149
= (egyenlő) 149
< (kisebb, mint) 150
> (nagyobb, mint) 150
!=(nem egyenlő) 150
<= (kisebb egyenlő) 150
>= (nagyobb egyenlő) 150
és 150
angtos 151
edtime 151
eq 152
eval 153
fix 153
getenv 153
getvar 153
if 153
index 153
linelen 154
nth 154
or 154
rtos 154
strlen 155
substr 155
upper 155
xor 155
Hibaüzenetek 155

6. FEJEZET ■ PROGRAMOZÁSI FELÜLETEK 157

Parancs-forgatókönyvek 158
A forgatókönyv elindítása az AutoCAD betöltésekor 158
Diabemutatók készítése 160
AutoLISP 160
AutoLISP alkalmazások használata 161
Automatikus betöltés és végrehajtás 162
Az *acad.lsp* fájl automatikus betöltése 162
Az *.mnl* fájl automatikus betöltése 164
Az automatikus betöltés parancsa 164
ADS—AutoCAD fejlesztői rendszer 165
Az ADS programok fejlesztésének feltételei 166
ADS alkalmazások használata 166
Az ADS alkalmazások automatikus betöltése 167
ARX—AutoCAD futásideje alatti kiterjesztése 168

I. RÉSZ
ALKALMAZÁSHOZ IGAZÍTÁSI REFERENCIA

A jelen részben az AutoCAD Release 13 program alapvető adaptációs lehetőségeit ismertetjük. Ezek a következők:

DOS programok és segédprogramok futtatása az AutoCAD programból. Lemezeket másolhat, vagy fájlokat törölhet le, feladatokat oldhat meg a megfelelő külső parancsnak a program paraméterfájlba (*acad.pgp*) történő hozzáadásával. A gyakran használt AutoCAD parancsok használatához rövidítéseket és álneveket hozhat létre. Például a DPFMÓD parancs meghívásakor elegedő a **dm** parancsot bebillentyűznie. A parancsálneveket a program paraméterfájlban definiálhatja (*acad.pgp*).

AutoCAD súgófájlok létrehozásával on-line súgót készíthet alkalmazásaihoz és saját eljárásaihoz. Ezek a fájlok mind az AutoLISP, mind az ADS programokban elérhetők. Az AutoCAD súgófájl szerkesztésével (*acad.ahp*) saját igényeihez igazított súgót hozhat létre az AutoCAD parancsokhoz ugyanúgy, mint saját alkalmazásaihoz.

Sraffozási mintákkal, alakokkal és szövegfontokkal kiegészített szaggatott és pontozott vonaltípusokat készíthet. Szintén létrehozhatja saját sraffozási mintáit, alakjait és szövegfontjait hozzáigazítva ezeket saját vállalata szabványaihoz és munkamódszereihez.

Saját menük létrehozásával igényeihez igazíthatja az AutoCAD felhasználói felületét. A menük az AutoCAD felületének számos részét vezérlik. Ha a legkönnyebben elérhető helyen, egy menüben helyezi el a legtöbbször használt parancsokat és ezeket csoportosítva egy egyszerű menüpontba rakja, megnövelheti munkájának hatékonyságát. A menük adaptálásával az AutoCAD felületét saját különleges szükségleteihez illesztheti. A nyomógombra kattintás eredménye a menüfájlban van definiálva, csakúgy mint legördülő, képernyő-, tábla- és

ábrazóna-menü működése és megjelenítése. Eszköztárakat és gyorsítóbillentyűket szintén a menüfájl segítségével adhat meg.

Az állapotsor adaptációja. Alapértelmezésben az AutoCAD állapotsora az aktuális fólianevet, a különböző módokat és a képernyő szálkeresztjének helyzetét mutatja. A DIESEL karakteres kifejezésnyelv segítségével és a MODEMACRO rendszerváltozóval igényeihez igazíthatja az állapotsort, így az képes lesz további AutoCAD és ADS információk (aktuális rajznév, rendszerváltozók beállítása) kijelzésére.

Automatikusan ismétlődő feladatok megoldása forgatókönyvek segítségével. A forgatókönyv egy olyan AutoCAD parancsokat tartalmazó ASCII szövegfájl, amelyet az AutoCAD a batch-fájlokhoz hasonlóan kezel, amikor lefuttat egy forgatókönyvet. Például, ha rajzokat kell kirajzolni, megírhat egy forgatókönyvet, ami megnyitja az összes rajzot, ki- és bekapcsolja a különböző fóliákat és kiadja a PLOT parancsot. Ezután éjszaka magára hagyva futtathatja a forgatókönyvet. A forgatókönyveket diákkal is használhatja automatizált és folyamatos bemutatók készítéséhez, ahogyan a kereskedelmi bemutatókon használatos. A dia egy nem szerkeszthető pillanatfelvétele az AutoCAD grafikus felületének. A diák emellett grafikus információk továbbadására is alkalmazhatók az ábrazóna-menükben és párbeszédpanelekben.

A kiválasztott AutoCAD parancsok átdefiniálása vagy letiltása, akár a parancspromptnál, akár egy AutoCAD vagy egy ADS program részeként. Az AutoCAD bizonyos parancsait újradefiniálhatja kiegészítő üzenetek vagy instrukciók megjelenítésére, vagy létrehozhat egy rajzkezelő rendszert, amelyben a KILÉP és VÉGE parancsokat újradefiniálhatja úgy, hogy a szerkesztett rész bezárása előtt a naplófájlba vezesse be a belépési és a kilépési időpontot.

Ez a referencia vezeti be az AutoCAD alkalmazási felületeinek (application programming interfaces, APIs) ismertetését is. Az AutoLISP, a LISP programozási nyelv egy különleges megvalósítása, beépített része az AutoCAD programnak. Az AutoLISP használatán keresztül az ismétlődő feladatok megoldásának automatizálásával és új AutoCAD parancsok definiálásával az AutoCAD programot munkájának igényeihez igazíthatja. Saját AutoLISP programokat is írhat, vagy használhatja a mások által megírt, illetve a shareware programokat. Az ADS (AutoCAD Development System) programozási felület segítségével a C programozási nyelvet is használhatja az AutoCAD program adaptálására. Mások által írt vagy saját ADS alkalmazásokat egyaránt futtathat.

1

AZ ADAPTÁLÁS ALAPJAI

Ez a fejezet tárgyalja az AutoCAD alapszintű adaptációs lehetőségeit. Ha nem rendelkezik kellő tapasztalattal e témakörben, érdemes elolvasnia ezt a fejezetet. Megtudhatja, hogyan lehet az AutoCAD programból az operációs rendszernek szóló parancsokat kiadni, vagy a gyakran használt parancsok helyett rövidítéseket létrehozni. A fejezet ezen kívül leírja azt is, hogyan készíthet on-line súgót saját alkalmazásaihoz.

Alapkövetelmények

A legtöbb, ebben a kézikönyvben leírt adaptációs eljárás megköveteli ASCII szövegfájlok létrehozását vagy módosítását. Ezért mindenképpen ismernie kell valamilyen egyszerűbb szövegszerkesztő programot (ilyen pl. az MS-DOS editor vagy a Windows™ Notepad) vagy fejlettebb szerkesztőrendszert (mint pl. a Microsoft® Word), amely képes ASCII formátumú fájlba elmenteni a szövegeket. Ezen kívül arról is meg kell bizonyosodnia, hogy nem felejtett el biztonsági másolatot készíteni az AutoCAD kiegészítő fájlokról még azok módosítása előtt.

Az AutoCAD környezete

Az AutoCAD program- és kiegészítő fájlok alapértelmezés szerinti könyvtárszerkezetét úgy tervezték, hogy az hatékony módon logikai csoportokba szervezze a fájlokat. Ha ez esetleg nem felel meg az igényeinek, akkor szabadon módosíthatja az alapbeállítást. Mivel néhány alkalmazás bizonyos fájlokat meghatározott helyeken keres, ellenőrizni kell, hogy a módosítások nem ütköznek ezen programok igényeivel. A meghajtó és a könyvtár nevének megadása nélkül az AutoCAD csak olyan fájlokat talál meg, amelyek az alább részletezett könyvtárkeresési útvonalon vannak.

A könyvtárkeresési útvonal

Az AutoCAD a kiegészítő fájlokat a *könyvtárútvonal* által meghatározott sorrendben keresi a következők szerint:

1 Aktuális könyvtár

2 Az aktuális rajzot tartalmazó könyvtár

3 A kiegészítő elérési útvonalban felsorolt könyvtárak

4 Az AutoCAD programfájlokat tartalmazó könyvtár

Megjegyzés Az aktuális környezettől függően ezen könyvtárak közül kettő vagy több megegyezhet.

Ha a fájl nem ezen a keresési útvonalon van, akkor ki kell jelölnie mind az elérési útvonal, mind a fájl nevét, hogy az AutoCAD megtalálhassa a fájlt. Például ha a *part5.dwg* nevű rajzot, ami nem a keresési útvonalon található, be akarja illeszteni saját aktuális rajzába, akkor meg kell adnia a teljes elérési útvonalat és a fájl nevét a következő módon:

Parancs: **Beill**
Blokk neve (vagy ?): **/files2/olddwgs/part5**

Ha a rajzfájl itt valóban megtalálható, akkor az AutoCAD felkéri Önt, hogy a szokásos módon folytassa a BEILL parancsot.

Megjegyzés Ha a Blokk neve (vagy ?) kérdésre a (~) karakter bebillentyűzésével válaszol, akkor meghívhatja a Rajzfájlok kiválasztása nevű párbeszédpanelt. Ez lehetővé teszi, hogy még abban az esetben is kiválaszthassa a fájlhoz tartozó könyvtárat és meghajtót, amikor nem ismeri a pontos fájlnevet és elérési útvonalat.

Kiegészítő fájlok elérési útvonala

Ez az elérési útvonal többféleképpen megadható. Az eljárások hierarchiája a következő (a magasabb sorszám hatástalanítja az alacsonyabb sorszámút):

1 Az ACAD környezeti változóval meghatározott könyvtárak

2 Az *acadenv.ini* fájlban meghatározott könyvtárak

3 Az *acad.ini* fájlban meghatározott könyvtárak

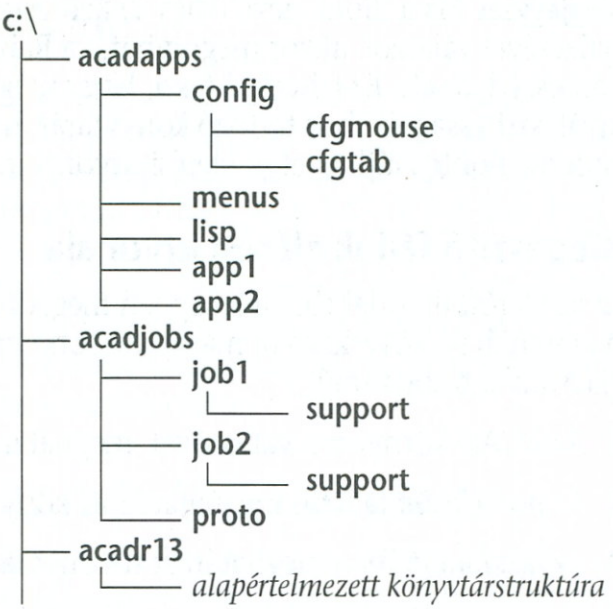
4 A \c parancssori kapcsoló után meghatározott könyvtárak (csak a Windows programban)

Az *AutoCAD Üzembehelyezési útmutató* negyedik fejezetében a 'Rendszerkörnyezet' címszónál megtalálhatja azokat az információkat, amelyek leírják a kiegészítő fájlok keresési útvonalának beállítását a fenti eljárások segítségével .

Egy példa könyvtár felépítésére

Ez a rész olyan lehetőségeket mutat be, amelyekkel Ön továbbfejlesztheti operációs rendszerének könyvtárát, és képessé teheti azt a fájlok könnyebb kezelésére. Az AutoCAD fastruktúrájú könyvtárszerkezetet használ. Ajánlatos a másodlagos fájlokat (mint az AutoLISP alkalmazások és menüfájlok) az AutoCAD fájljaitól és a kiegészítő fájloktól elkülöníteni. Így könnyebb megtalálni a lehetséges problémákat, és egy-egy alkalmazást a többitől függetlenül is tovább lehet fejleszteni.

A további fejezetek a következő ábrán látható minta-alkönyvtár rendszerre hivatkoznak, mint az alkönyvtárak szervezésének módszerére.



Egy példa az AutoCAD könyvtárstruktúrájára

Létrehozhatja saját alkönyvtárát is saját AutoLISP alkalmazásai, menüfájljai, és egyéb, mások által megírt alkalmazások számára (mint az előző példában az */acadapps* könyvtár). Ha többszörös rajzkönyvtárakat kíván fenntartani (a munkafájlok elkülönítésére), a */acadjobs* könyvtárhoz hasonló könyvtárakat hozhat létre rajztípusonkénti alkönyvtárakkal.

Többféle konfiguráció

Amikor az AutoCAD programot (például az AutoCAD Telepítési útmutató 2., "Az AutoCAD konfigurálása" című fejezetében található módon) konfigurálja, az Ön által megadott információk az *a*cad.cfg konfigurációs fájlban kerülnek rögzítésre. A rendszerkönyvtárak meghatározása után az AutoCAD a konfigurációs könyvtárat határozza meg. Ez a két könyvtár gyakran azonos, és amennyiben csak egyféle konfigurációra van szüksége, akkor ez az alapbeállítás tökéletesen megfelel.

Általában elegendő egyetlen konfiguráció, de néha többféle konfigurációra is szüksége lehet. Ha például munkáinak többségéhez egeret használ, de alkalmanként szüksége van egy nagyméretű digitalizáló táblára, felépítheti úgy is a rendszerét, hogy az több konfigurációt tudjon kezelni, és ne legyen szükséges minden alkalommal újrakonfigurálnia, amikor eszközt vált.

Az *acad.cfg* fájl számos AutoCAD rendszerváltozó értékét és a Működési paraméterek menü konfigurációs opcióit is tárolja. Amennyiben e változóknak és működési paramétereknek különböző beállításaira van szüksége, az azoknak megfelelő acad.cfg fájlokat elmentheti különböző könyvtárakba. Az AutoCAD on-line súgója a

"Rendszerváltozók" címszó alatt felsorolja valamennyi rendszerváltozót, valamint tárolásuk helyét.

A többféle konfigurációval járó előnyök kiaknázásához Önnek először külön lemezkönyvtárakat kell létrehoznia, amelyek a különböző acad.cfg fájlokat fogják tartalmazni. A minta könyvtárstruktúra a konfigurációs könyvtárak egyik lehetséges elrendezését mutatja be. Egy konfigurációs könyvtárat az ACADCFG környezeti változó megadásával hozhat létre. A különböző rendszerek ennek a változónak a beállítását különbözőképpen kezelik, azonban az eredmény ugyanaz. DOS és UNIX rendszerekben batch- vagy forgatókönyvfájl formájában adhatja meg a konfigurációs könyvtárat. Windows rendszerben a Programkezelő (Program Manager) csoportablakban minden egyes konfigurációnak külön programikont készíthet és a /c parancssori kapcsolóval megadhatja a konfigurációs könyvtárat.

Ha az AutoCAD az *acad.cfg* fájlt nem találja a megadott könyvtárban, újra kell konfigurálnia az AutoCAD programot. Amikor a konfigurációs eljárást befejezte, az AutoCAD az ACADCFG környezeti változó által megadott könyvtárban létrehoz egy *acad.cfg* fájlt. Ezt az eljárást minden egyes konfigurálásnál meg kell ismételnie.

Többféle rajzkönyvtár

Több rajzkönyvtár fenntartása általában nemcsak kényelmes, hanem gyakran szükséges is. Rajzainak és más, azokkal kapcsolatos fájloknak különálló könyvtárakban történő tárolása jóval egyszerűbbé teszi a fájlok alapvető karbantartását. Az alábbiakban leírt módszer a korábban tárgyalt minta-könyvtárstruktúrára alapul, de a saját igényei szerint szabadon átalakíthatja, illetve továbbfejlesztheti.

Rajzait tartalmazó alkönyvtárai számára létrehozhatja a */acadjobs* könyvtárat. Alkönyvtárain belül további alkönyvtárakat tarthat fenn, amelyekben az egyes rajztípusokhoz vagy munkákhoz tartozó kiegészítő fájlok helyezkedhetnek el. Például a */acadjobs/job1/support* alkönyvtár tartalmazhat blokkokat és AutoLISP fájlokat, amelyek az */acadjobs/job1* alkönyvtárban elhelyezkedő rajzfájlokhoz tartoznak. A **support** (elérési útvonal nélküli) megadásával az aktuális alkönyvtáron belüli *support* alkönyvtárat adhatjuk hozzá a support keresési útvonalhoz.

Ha megbizonyosodott róla, hogy a kívánt rajzkönyvtár az aktuális az AutoCAD indulásakor, akkor az abban található összes fájl és alkönyvtár a későbbiek során könnyedén elérhető. Ha DOS vagy UNIX operációs rendszert használ, akkor aktualizálhatja a kívánt könyvtárat a **cd** paranccsal, mielőtt az AutoCAD programot indító batch fájlt vagy makrót lefuttatja. Ellenőriznie kell azt is, hogy a batch fájl ne adjon meg új aktuális könyvtárat. Windows alatt minden egyes rajzkönyvtárhoz egy külön ikont adhat meg a programkezelő

csoportban. Az ikon létrehozásánál munkakönyvtárként az adott rajzkönyvtár nevét adja meg.

Prototípus-rajzok

A prototípus-rajzokat kiindulási alapnak használhatja, amikor egy új rajzhoz kezd hozzá. Ha hatékonyan kihasználja ezt a lehetőséget, azzal jelentős időt és energiát takarít meg. Valamennyi meglévő rajzot felhasználhatja prototípus-rajzként. A prototípus-rajzok minden olyan információt tárolnak, amelyet egy rajzfájl tartalmaz. Amikor prototípusként használ egy rajzot, akkor az új rajz az abban tárolt összes információt átveszi.

A prototípus-rajzok segítségével biztosítható, hogy az Ön szabvány rajzainak megfelelő fóliák és alapelemek rendelkezésre álljanak. Mivel egy adott prototípus-rajzot igen könnyű kiválasztani, külön prototípusai lehetnek minden rajzlapmérethez, méretarányhoz vagy rajzfajtához. Az ÚJ parancs kiadásakor az Új rajz létrehozása párbeszédpanel jelenik meg, amelyben a Prototípus... nyomógombra rákattintva lehetősége nyílik kiválasztani azt a prototípus-rajzot, amelyet alapul kíván venni az új rajzhoz.

Az alapértelmezés szerinti prototípus-rajzot az Új rajz létrehozása párbeszédpanelben vagy a Konfigurációs menü Működési paraméterek konfigurálása opciójában, a Kiinduló rajzbeállítás kiválasztásával adhatja meg. Az így megadott alapértelmezés szerinti prototípus-rajzot tartalmazó könyvtár lesz az alapértelmezés szerinti könyvtár az Új rajz létrehozása párbeszédpanel Prototípus... nyomógombjának megnyomásakor. A */acad/support/proto* könyvtár (ahogyan a minta könyvtárstruktúrában látható) alapértelmezés szerinti prototípus rajzkönyvtárként használható.

Parancskeresési eljárás

Egy parancs bebillentyűzésekor, az AutoCAD több lépésben vizsgálja meg a parancsnév érvényességét. A parancs lehet belső parancs vagy az *acad.pgp* fájlban meghatározott rövidített álnév parancs, vagy a felhasználó által definiált parancs. Parancsokat ADS alkalmazások illetve képernyő- vagy eszközmeghajtó parancsok is definiálhatnak. A parancsot beírhatja a parancssorba, vagy kiválaszthatja a megfelelő menüből. A parancsok AutoLISP vagy ADS alkalmazások forgatókönyvfájljaiból is betölthetők.

A következő lépések bemutatják az AutoCAD parancskereső eljárását:

1 Ha a bemenet egy üres válasz billentyű (Spacebar vagy ↵), az AutoCAD az utoljára kiadott parancs nevét használja fel. A SÚGÓ az alapértelmezés szerinti parancs.

2 Az AutoCAD ellenőrzi a nevet a beépített parancsok listáján. Ha a név szerepel a listán, és előtte nincsen pont, a rendszer ellenőrzi a

nevet a nem definiált parancsok listáján. Ha a parancs nem definiált, a keresés folytatódik; egyébként végrehajtódik, hacsak más okból ez nem lehetséges (például transzparens parancsként vagy perspektív módban sikertelennek bizonyulhat a futtatás).

3 Az AutoCAD ellenőrzi a nevet az eszközmeghajtó, majd a képernyőmeghajtó által definiált parancslistákban.

4 Az AutoCAD ellenőrzi a nevet a programparaméter-fájlban (*acad.pgp*) definiált külső parancsok között. Ha a név megfelel egy már definiált külső parancs nevének, akkor a parancs végrehajtása elkezdődik és a keresés befejeződik.

5 Az AutoCAD ellenőrzi a nevet a felhasználó által definiált ADS és AutoLISP függvények listájában. Ez az a pont, ahol egy automatikusan betöltődő parancs betöltődik. (Az automatikusan betöltődő parancsokról az "Az automatikus betöltés parancsa" című részнél, a 164. oldalon olvashat.)

6 Az AutoCAD ellenőrzi a nevet a rendszerváltozók között. Ha megtalálja, akkor az AutoCAD végrehajtja a VÁLTOZÓK parancsot, az adott nevet használva változónévként.

7 Ha a parancsnév megtalálható a programparaméter-fájlban, a rövidített parancsálnevek között, az AutoCAD a kibővített parancsnevet használva folytatja a keresést a 2. ponttól kezdődően.

8 Ha az összes eddigi lépés eredménytelen, a keresés megszakad, és egy érvénytelen parancsnévre vonatkozó üzenet jelenik meg.

Módosítható kiegészítő fájlok

Az AutoCAD kiegészítő (support) fájlokat használ olyan célokra, mint a menük definícióinak tárolása, AutoLISP és ADS rutinok betöltése vagy szövegfontok definiálása. A kiegészítő fájlok között sok a szöveges fájl, melyeket szövegszerkesztőben módosíthat.

Az alábbi lista azokat az AutoCAD kiegészítő fájlokat tartalmazza, amelyek szövegformátuma editálható. Javasoljuk, hogy mielőtt módosítaná ezeket a fájlokat, készítsen róluk biztonsági másolatot.

Módosítható kiegészítő fájlok

Fájl	*Leírás*
acad.ads	Ez a fájl azoknak az ADS alkalmazásoknak a felsorolását tartalmazza, amelyeket az AutoCAD induláskor automatikusan betölt
**.ahp*	AutoCAD súgófájlok. A hozzájuk tartozó súgó indexfájlok kiterjesztése *.hdx*
acad.ahp	Az *acad.ahp* rendszerfüggetlen AutoCAD súgó fájl

Fájl	*Leírás*
acad.ase	Ez a fájl közli az AutoCAD programmal az AutoCAD SQL Kiterjesztés (ASE) adatbázismeghajtóinak nevét és helyét
* *.dcc*	(Csak DOS esetében) Ezek a fájlok tartalmazzák az AutoCAD párbeszédpanelek színének beállítását
* *.dcl*	Ezek a fájlok tartalmazzák a párbeszédpanelek leírását az AutoCAD Párbeszédvezérlő nyelven (Dialog Control Language, DCL)
acad.dcl	Az *acad.dcl* tartalmazza az AutoCAD párbeszédpaneleinek leírását
acad.ini	AutoCAD inicializáló fájl. Általános konfigurációs és setup-információkat tartalmaz.
acadenv.ini	AutoCAD kiegészítő, környezeti inicializáló fájl. Azon könyvtárak elérési útvonalait tartalmazza, amelyek az AutoCAD környezetét alkotják. (Ha nincsenek megadva az *acad.ini* fájlban).
.lin	AutoCAD vonaltípusok definíciós fájljai
acad.lin	Az *acad.lin* tárolja az AutoCAD vonaltípusainak definícióját
*.lsp	AutoLISP programfájlok
acad.lsp	Az *acad.lsp* fájl egy, a felhasználó által definiált AutoLISP rutin, melyet az AutoCAD automatikusan betölt valahányszor új rajzba kezdünk
*.mln	Vonal könyvtárfájl.
*.mnd	A lefordítatlan menüfájlok speciális típusa, amely makrókat tartalmaz. Az mc futtatható fájl használható ennek a fájlnak a lefordítására
*.mnl	Ezek a fájlok tartalmazzák az AutoCAD menükben használt AutoLISP rutinokat. Minden *.mnl* fájl nevének meg kell egyeznie a hozzá tartozó *.mnu* fájléval
acad.mnl	Az acad.mnl tartalmazza az AutoCAD menü által használt AutoLISP rutinokat
*.mns	(csak Windows programban) Az AutoCAD által generált menü forrásfájlok. Az AutoCAD menüket leíró parancssorokat és makró szintaktikát tartalmazzák.
acad.mns	A szabványos AutoCAD menü forrásfájlja.
*.mnu	AutoCAD forrás menüfájlok. Ezek tartalmazzák az AutoCAD menüket definiáló parancssorozatokat és makrónyelvi elemeket.
acad.mnu	Az *acad.mnu* fájl az AutoCAD menü forrásfájlja

Fájl	*Leírás*
acad.msg	AutoCAD üzenetfájl. Az AutoCAD NÉVJEGY parancsa az ebben a fájlban található információt jeleníti meg.
**.pat* *acad.pat*	AutoCAD sraffozási minták definíciós fájljai Az *acad.pat* fájl tárolja az AutoCAD sraffozási mintáinak definícióját
**.pcp*	AutoCAD kirajzoltatási konfigurációs paraméterfájlok. Minden egyes *.pcp* fájl egy meghatározott plotterre vonatkozó konfigurációs adatokat tartalmaz
acad.pgp	AutoCAD programparaméter-fájl. Ez tartalmazza a külső parancsok és a parancsálnevek meghatározását
fontmap.ps	AutoCAD Fonttáblafájl, melyet a PSBE parancs használ. Ez az AutoCAD PostScript értelmező által ismert összes font katalógusa (vagy megfeleltetési táblázata)
acad.psf	AutoCAD PostScript Support fájl. Ez a PSKI és PSKITÖLT parancsok Master Support alapfájlja
**.scr*	AutoCAD forgatókönyvfájlok. Minden forgatókönyvfájl AutoCAD parancsok sorozatát tartalmazza, melyek automatikusan (batch módon) kerülnek végrehajtásra
**.shp*	AutoCAD Alak-/Fontdefiníciós fájlok. A lefordított Alak-/Fontfájlok kiterjesztése *.shx*
acad.unt	AutoCAD mértékegység-definíciós fájl. Ez tartalmazza a mértékegységek közötti átszámításhoz szükséges adatokat

Programparaméter-fájl—*acad.pgp*

Az AutoCAD *acad.pgp* nevű programparaméter-fájlja olyan ASCII szövegfájl, amely parancsok definícióit tárolja. Az alkalmazott AutoCAD parancsok listájának tekinthető. Amikor begépelünk egy olyan utasítást, amely nem eredeti AutoCAD parancs, akkor a rendszer azt ebben a fájlban keresi. Ez a fájl két részből tevődik össze: az első rész külső parancsokat, a második pedig parancsálneveket definiál. Ezenkívül a fájl megjegyzéseket tartalmazhat, amelyeket pontosvessző előz meg.

Minden egyes alkalommal, amikor új vagy már létező rajzot nyit meg, az AutoCAD megkeresi a megfelelő könyvtárútvonalat, és beolvassa az első *acad.pgp* fájlt, amit talál. Ha a szerkesztés során újra akarja inicializálni (tölteni) az *acad.pgp* fájlt, használja az ÚJRAINIC parancsot.

Külső parancsok definiálása

Az AutoCAD futása közben más, az alább felsoroltakhoz hasonló programokat és segédprogramokat is elindíthat:

- DOS parancsok és segédprogramok mint például a **type**, **dir**, vagy **copy**
- Editorok és szövegszerkesztők
- Adatbáziskezelők, táblázatkezelők, kommunikációs programok
- Felhasználók által készített prgramok, például batch-fájlok

Amikor egy külső parancsot definiál (például a SHELL parancsot), minden egyes sor egy-egy, az AutoCAD Parancs promptjából elindítható parancsot definiál. A külső utasítások minden egyes sora öt, vesszővel elválasztott mezőből áll, melyek a következők:

```
parancs,végrehajtható,mem[,prompt[,visszatérési_kód]]
```

Parancs A parancsot a parancssorban kell megadni. Ha ez a név megegyezik egy belső AutoCAD parancs nevével, akkor ezt a parancsot a program nem veszi figyelembe. A névben nincs különbség kis- és nagybetű között.

végrehajtható Amikor a parancs nevét beírja, ez a konstans karakterlánc kerül az operációs rendszerbe. Ez bármilyen parancs lehet, amely az operációs redszer promptjától végrehajtható. A karakterlánc kapcsolókat és paramétereket is tartalmazhat. Az Ön operációs rendszerétől függ, hogy a karakterláncban meg kell-e különböztetni a kis- és nagybetűket.

mem Ez a mező továbbra is kompatibilis az AutoCAD korábbi verzióival. Bár az AutoCAD nem használja a mezőben található értékeket, mégis kell itt állnia valamilyen számnak (ez általában 0).

prompt Ez a mező tetszőlegesen választható. Egy promptot jelenít meg az AutoCAD parancssorában. Az erre a promptra adott választ az AutoCAD a végrehajtható mezőben megadott konstans karakterlánchoz illeszti. Abban az esetben, ha a mező első karaktere egy csillag jel (*), akkor a válasz szóközöket is tartalmazhat, és csak a ↵ billentyű lenyomásával fejeződik be. Máskülönben a választ a Spacebar vagy a ↵ billentyűvel zárhatja le. Ha nem kíván promptot megjeleníteni, nem kell bebillentyűznie semmit;

viszont ki kell tennie egy vesszőt, amennyiben a visszatérési kód meg van adva.

visszatérési kód

Ez egy tetszőlegesen választható, bitkódként értelmezett paraméter. Ezeket az egész szám értékeket bármilyen sorrendben összeadhatja a kívánt eredmény eléréséhez. Például ha 2 és 4 értékeket kell megadni, a visszatérési kód értéke 6. Az egyes értékek definíciója a következő:

0 *Visszatérés a szöveges képernyőhöz*

A képernyő szöveges módban marad a parancs befejezése után, kivéve akkor, ha DXB fájl betöltése miatt az AutoCAD grafikus üzemmódba tér át.

1 *DXB fájl betöltése*

A parancs végeztével az AutoCAD betölti a rajzba a *$cmd.dxb* nevü DXB fájlt. Amint a DXB fájl betöltődött, a *$cmd.dxb* fájl törlődik. Az eredmény ugyanaz, mint a DXBBE parancs végrehajtásakor.

2 *Blokkdefiníció létrehozása DXB fájlból*

Az AutoCAD létrehoz egy blokkdefiníciót a *$cmd.dxb* nevü DXB fájlból. A prompt mezőre adott válasz szolgáltatja a blokk nevét. Ennek a névnek olyan érvényes blokknévnek kell lennie, amely az aktuális rajzon nem létezik. Ezért ebben a módban nem lehet korábban definiált blokkokat újra definiálni. Miután az AutoCAD betölti a DXB fájlt, a *$cmd.dxb* fájl törlődik. A BEILL parancs alapértelmezés szerinti neve az új blokk neve lesz.

4 *Visszatérés az előző képernyőmódhoz*

A parancs végeztével az AutoCAD visszaállítja az előző képernyőmódot (ez legtöbb esetben grafikus üzemmód).

A következő példa két új külső parancsot definiál: a LISTSET és a DXB2BLK parancsot.

```
CATALOG,DIR /W,0,Fájlspecifikáció: ,0
DEL,DEL,0,Törölni kívánt fájl: ,4
DIR,DIR,0,Fájlspecifikáció: ,0
EDIT,EDLIN,0, Szerkeszteni kívánt fájl: ,4
SH,,0,*OS Parancs: ,4
SHELL,,0,*OS Parancs: ,4
TYPE,TYPE,0,Listázni kívánt fájl: ,0
LISTSET,SET,0,
DXB2BLK,DXBCOPY,0,DXB fájl blokkosítása: ,3
```

A LISTSET parancs az aktuális DOS környezeti változó értékeit jeleníti meg.

A DXB2BLK parancs a megadott DXB fájlból egy blokkdefiníciót hoz létre. A DXB fájl minden objektumot sorrá konvertál. Az eljárás egy hasznos melléktermékeként egyszerű megoldást nyújt szöveg-objektumok sorokra történő tördelésére.

A DXB2BLK az adott DBX fájl nevét a *dxbcopy* batch fájlba küldi, amely a fájl nevét a *$cmd.dxb* nevű fájlba másolja át. Az AutoCAD ekkor az adott DXB fájlból egy blokkot hoz létre. Az új blokk neve a DXB fájl promptnál megadott név. A *dxbcopy.bat* fájl létrehozásához a következő parancsot adja meg DOS promptnál:

```
echo copy %1.dxb $cmd.dxb > dxbcopy.bat
```

Ez az aktuális könyvtárban létrehozza a *dxbcopy.bat* fájlt. Tegye át ezt a fájlt az Ön saját DOS könyvtárába, vagy pontosan adja meg a fájl helyét az *acad.pgp* fájlban. Például ha a *dxbcopy.bat* fájl a D:/*cad* könyvtárban van, a következőket adja meg az *acad.pgp* fájl külső parancsaként.

```
DXB2BLK,D:\CAD\DXBCOPY,0,DXB fájl blokkosítása: ,3
```

DXB fájlt létrehozhat úgy, hogy a Plotter Konfigurálás menüben az AutoCAD fájl kimeneti formátumaként AutoCAD DXB fájlt ad meg. Erről bővebben az AutoCAD Üzembehelyezési útmutató 2., 'Az AutoCAD konfigurálása' című fejezetében olvashat.

Parancsálnevek

A gyakran használt AutoCAD parancsokat lerövidítheti, ha az *acad.pgp* fájl parancsálnevekre vonatkozó részében álnevet ad nekik. Bármely AutoCAD parancsnak, eszközmeghajtó-parancsnak vagy külső parancsnak adhat álnevet.

Az *acad.pgp* fájl következő két, egymástól vesszővel elválasztott mezőjében adhatja meg a parancs álnevét:

```
rövidítés,*parancs
```

rövidítés Ez a parancssorba írandó parancs rövidítése.

parancs Ez a lerövidíteni kívánt AutoCAD parancs. A parancs elé egy csillagot (*) kell írnia, mert az AutoCAD erről tudja azonosítani a parancsálnevet tartalmazó sort.

A parancsálnevet is használhatja transzparens módon, amennyiben az eredeti parancs esetében ez lehetséges. Amikor parancsálnevet ad meg, az AutoCAD a teljes parancsnevet kijelzi a parancssornál, és végrehajtja a parancsot.

Íme egy parancsálnév részlete, mely az *acad.pgp* fájlban találhatókhoz hasonló tételeket tartalmaz.

```
I,*ÍV
K,*KÖR
M,*MÁSOL
```

A parancsálnév szakasz mínusz jelet (-) is tartalmazhat. Ez lehetővé teszi , hogy olyan álneveket hozzon létre, amelyek elérik bizonyos parancsok parancssorbeli változatát.

```
BH, *-BHATCH
BD, *-BOUNDARY
```

Megjegyzés Parancsforgatókönyvekben nem használhat parancsálneveket. Használatuk a menüfájlokban nem ajánlott.

Saját súgófájlok

Az AutoCAD lehetővé teszi, hogy a súgót adaptálja az AutoCAD utasításaihoz, promptjaihoz és egyéb szolgáltatásaihoz csakúgy, mint a saját alkalmazásaihoz. Az AutoCAD súgófájl-formátuma minden számítógép-platformon rendelkezésre áll. Módosíthatja az eredeti AutoCAD súgófájlt, *acad.ahp* vagy saját fájlt hozhat létre. Ezeket a fájlokat az AutoLISP és az ADS alkalmazások is képesek meghívni. Az AutoCAD súgófájlok ASCII szövegformátumúak , és *.ahp* kiterjesztésűek. Ha Windows alatt használja az AutoCAD programot, használhatja a Windows súgófájlokat, azonban nem módosíthatja az *acad.hlp* fájlt, újat kell írnia.

Alapértelmezésben a SÚGÓ parancs az AutoCAD párbeszédpaneljét használja az *acad.ahp* fájl kijelzésére bármely környezetben, kivéve a Windowst. A Windows rendszerben a SÚGÓ parancs a Windows súgóját használja az *acad.hlp* fájl kijelzésére.

Az AutoCAD minden egyes AutoCAD súgófájl (*.ahp*) részére egy indexfájlt (*.hdx*) hoz létre és tart fenn, hogy a súgó tökéletesen működjék. Ha megváltoztat egy súgófájlt, ezt az indexet az AutoCAD aktualizálja, amikor legközelebb a SÚGÓ parancs kiadásra kerül. Miután módosított egy súgófájlt, törölje ki a létező *.hdx* fájlt, hogy biztosítsa az index aktualizálódását.

Az AutoCAD súgófájl formátuma a Relese 12 verzióhoz képest megváltozott. A következő rész tartalmazza az új formátum leírását. Ha már meglévő Release 12 súgófájlokat új formátumra kíván konvertálni, olvassa el az "R12 súgófájlok konvertálása R13 súgófájlokká" című résznél, a 34. oldalon található ismertetést.

Az AutoLISP és az ADS alkalmazások a **help** függvény segítségével képesek egyéb súgófájlok megjelenítésére is. A **help** függvény leírását a 13. fejezetben a "**help**" címszó alatt és a "Súgófájlok elérése" című résznél, a 255. oldalon találja meg. Az **ads_help()** függvény leírását megtalálja az *AutoCAD Developers Guide (AutoCAD Fejlesztõi kézikönyv)* 3. fejezetében. Súgófájlokat az AutoLISP **help** függvényének

parancssorban való kiadásával is megjeleníthet. A következő kód az AutoCAD *ensugom.ahp* súgófájlt jeleníti meg.

Parancs: **(help "ensugom.ahp")**

Windows környezetben az AutoCAD a menüfájlok segítségével további súgó lehetőségeket kínál, erről információt a "Menü-specifikus súgó" című résznél, a 134. oldalon talál.

Az AutoCAD súgófájl formátuma

Az AutoCAD súgófájl szerkezete hasonló a Windows súgófájléhoz. Az AutoCAD súgófájlban az információkat téma szerint vannak csoportosították. Minden témának egyedi azonosítója kell legyen, melyet témaazonosítónak (néha kontextus-karakterláncnak) hívnak. Megadhatja mind a téma címét, mind a hozzátartozó kulcsszavakat. A témaazonosító, a cím és a kulcsszavak külön sorokban, un. *direktívákban* vannak megadva. Minden direktívasor első karaktere egy visszahajló törtjel (\). A következő táblázat tartalmazza az AutoCAD súgódirektíváit (az első három direktívának \#, \$, és \K a megadott sorrendben kell állnia).

AutoCAD súgódirektívák

Direktíva	*Használata*	*A kijelzés helye*
\#	témaazonosító (kontextus-karakterlánc)	A felhasználó nem látja; belső felhasználású; egyedi kell legyen; nincs szóköz
\$	Cím	Előző parancsok és a keresési témák listája
\K	Kulcsszó	Keresési lista. A kulcsszavakat pontosvessző (;) választja el.
\E	Fájl vége	A súgófájl vége; csak egyszer, a fájl végén használandó
\(szóköz)	Megjegyzés	Azokat a sorokat, melyek első karaktere visszahajló törtjel (\), és azt szóköz követi, a program nem veszi figyelembe.

Az AutoCAD súgófájlok direktíváira vonatkozó szabályok a következőek:

- minden témának kell legyen kontextus-karakterlánca, az ilyen azonosítóval nem rendelkezõ témákat, melyeknek nincs, a program nem veszi figyelembe.
- A téma befejeződik, ha a program egy új direktívával találkozik.

- A direktíva-karakterlánc közvetlenül a \#,\$,\K direktívák után, a harmadik karakterpozícióban kezdődik (ha szóköz követi a direktívát, a szóköz része a direktíva-karakterláncnak)

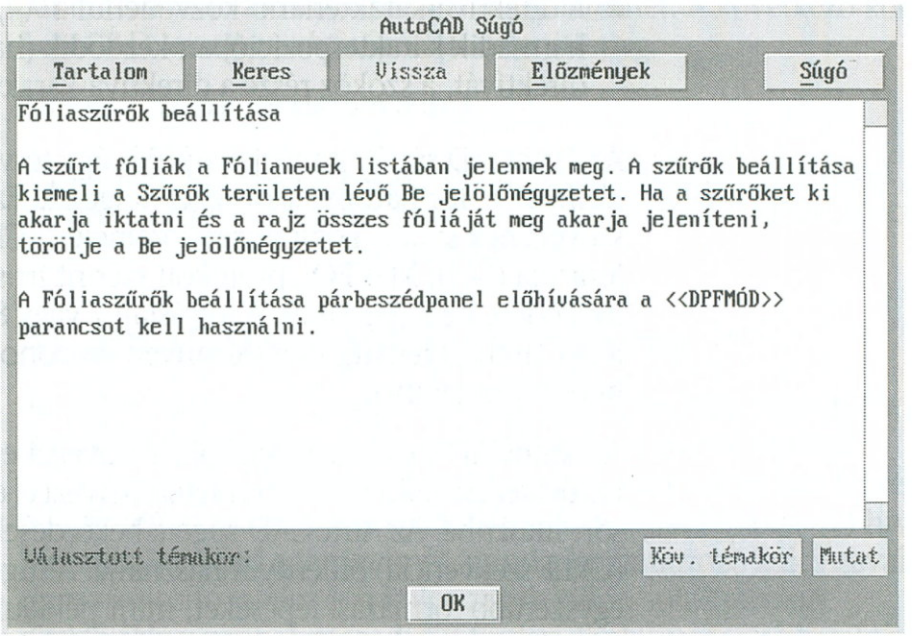
Az AutoCAD súgófájl megjelenítőjét úgy tervezték, hogy DOS vagy UNIX EOL formátumú szöveggel dolgozik. Ezért a súgó forrásfájl szövegének sorait "kocsi vissza-soremelés" (CRLF) billentyűnek kell lezárnia (↵). Más EOL protokoll szerint írt súgófájlok tökéletes működése nem garantált. A forrásfájl 7 vagy 8 bites ASCII formátumú lehet, ami azt jelenti, hogy könnyen beazonosítható bármely, nem 16 bites környezetben.

Az AutoCAD súgó megjelenítője a szöveget egy legördülő listaablak rendelkezésre álló munkaterületén helyezi el és a bekezdések közé üres sort illeszt be. Az AutoCAD súgó a bekezdések elválasztására a ↵ (CRLF szekvencia) billentyűt használja. A súgó megjelenítője kezeli az egyszerűbb formázási lépéseket, mint például a soft return, a tabulátorok és a bekezdések (További információ található a "Speciális formázás" című részнél, a 32. oldalon).

A következő példa egy AutoCAD *acad.ahp* súgófájlban található témát mutat be. Mivel a ↵ (CRLF szekvencia) helyét fontos megjeleníteni, ezért ennek jelzésére a példa a ¶ karaktert használja a ↵ helyén.

```
\#set_layer_filters_dialog¶
\$Fóliafiltereket beállító párbeszédpanel¶
\KDPFDÓD;párbeszédpanelek;Fóliafiltereket beállító
párbeszédpanel¶
Fóliafilterek beállítása¶
A kiszűrt fóliák a Fólianevek listájában jelennek meg. A
filterek megadásához válassza ki a BE ablakot a Filterek
területről.Szüntesse meg a kiválasztást a filterek
kikapcsolásához és a rajzban az összes fólia megjelenik.¶
A Fóliafiltereket beállító párbeszédpanel eléréséhez
használja a <<DPFMÓD>>dpfmód> parancsot.¶
\#select_color_dialog_box¶
```

A téma egyedi azonosítója a `set_layer_filters_dialog`. A címe `Fóliafiltereket beállító párbeszédpanel`, és három kulcsszóval lehet elérni: `KDPFDÓD, párbeszédpanelek`, és `Fóliafiltereket beállító párbeszédpanel`. A téma alapszövegén belül három CRLF szekvencia jel található, amelyek három bekezdésre bontják a szöveget. A téma tartalmaz egy hipercsatolást a DPFMÓD parancshoz (további információ a "Hiperszöveg-csatolások" című részнél, a 33. oldalon). A témát a `\#select_color_dialog_box` direktíva zárja, amely egyben az új témát kezdi. A következő ábra megmutatja, hogy az első téma hogyan jelenik meg az AutoCAD súgómegjelenítőjén.



Speciális formázás

Az AutoCAD súgómegjelenítője lehetővé teszi, hogy finomításokat végezzen el a téma szövegén. A bekezdések közötti üres helyeket a visszahajló törtjel (\) jelzéssel, (mint egy soft return) tudja megszüntetni. A program a tabulátorjelzéseket felismeri és tabulátorpontokat tesz minden negyedik karakterpozícióra. A súgómegjelenítő a bekezdéseket úgy illeszti egymáshoz, hogy a szöveget kihúzza az utolsó tabulátorpozícióra. Ez a formázás megfelelően kezeli a pontozott és számozott bekezdéseket, de egy egyedi megoldásra van szükség, ha a címke hosszabb néhány karakternél. Az alábbi súgófájl kód bemutatja ezeket a speciális formázási lehetőségeket. A jobbra nyíl (→) a tabulátort jelzi.

```
\#sample_topic¶
\$Példatéma¶
\KPélda;Teszt¶
Példatéma (szóköz alul)¶
Ez az első példa\¶
megmutatja a "soft return" használatát (nincs szóköz
felül)¶
→Ez a példa bemutatja, hogyan hozhatunk létre bekezdéssel
egy paragrafust. Mivel ez egy tabulátorral kezdődik az
egész paragrafus beljebb lesz tolva.¶
→1.→Ez a példa a sorszámozott listák készítésének
technikáját mutatja be. Megismerheti ennek a jellemzőnek
a használatát a "függő bekezdések" létrehozásával
kapcsolatban¶
NAGY CÍMKE→Ez a példa azt a módszert mutatja be, ami akkor
szükséges,\¶
→→→ha olyan címkéket akar használni, amelyek hosszabbak
egy karakternél. Ez számos példa és hiba tesztelésénél, a
helyes eredmény elérésében lehet hasznos.¶
\E¶
```

A következő példa azt mutatja be, hogy az előző kód hogyan jelenne meg az AutoCAD súgómegjelenítőjén.

```
Példatéma (szóköz alul)

 Ez az elsõ példa
megmutatja a "soft return" használatát (nincs szóköz
felül)
    Ez a példa bemutatja, hogyan hozhatunk létre
bekezdéssel egy paragrafust. Mivel ez egy tabulátorral
kezdődik az egész paragrafus beljebb lesz tolva.
    1. Ez a példa a sorszámozott listák készítésének
technikáját mutatja be. Megismerheti ennek a jellemzőnek
a használatát a "függő bekezdések" létrehozásával
kapcsolatban.
  NAGY CÍMKE Ez a példa azt a módszert mutatja be, ami
akkor szükséges, ha olyan címkéket akar használni, amelyek
hosszabbak egy karakternél. Ez számos példa és hiba
tesztelésénél, a helyes eredmény elérésében lehet hasznos.
```

Hiperszöveg-csatolások

Létezhetnek hiperszöveg-csatolások különböző témák között, egyazon súgófájlon belül. A hiperszöveg-csatolások fajtái: témaugrás vagy téma-felülírás (pop-up). A hiperszöveg-csatolás szintaxisa a következő:

```
<<hot spot>>téma_az>
<<hot spot>>téma_az]
```

A `téma_id-t` követő > ("nagyobb mint" jel) témaugrást, a] (jobb zárójel) téma-felülírást jelöl. Mindkét esetben a `<<hot spot>>` jelenik meg a súgómegjelenítőben. A `téma_az>` és a `téma_az]` a felhasználó számára nem látható. Amikor a felhasználó kettőt kattint a görgethető listaablak témaugrás szövegének egy olyan sorára, amely egy `<<hot spot>>`-ot tartalmaz, a megjelenítő azt a témát jelzi ki, amely a `téma_az` azonosítóhoz kapcsolódik. Ha a felhasználó a téma-felülírás `<<hot spot>>` azonosítót választja a `téma_az`-hoz kapcsolódó téma egy olyan párbeszédpanelben jelenik meg, mely a súgómegjelenítőt felülírja. Amikor a téma-felülírás párbeszédpanelja eltűnik, a súgómegjelenítő az eredeti témát mutatja.

Ha a `<<hot spot>>` egy másik sorhoz kapcsolódik, akármelyik sor kiválasztható. Ha egy sorban egynél több `<<hot spot>>` létezik, a felhasználó választhatja ki a számára megfelelő témát a Következő és a Mutat billentyűkkel.

R12 súgófájlok konvertálása R13 súgófájlokká

Elképzelhető, hogy Release 12 verzió súgófájljait a Release 13 verzióban akarja használni. Ha a már létező súgófájljait fejleszteni kívánja, akkor bizonyára időt szán arra, hogy további kulcsszavakat és hipercsatolásokat hozzon létre. Ezt kezdheti a következőkben ismertetett eljárás végrehajtásával. Az eljárás vázolja azokat a lépéseket, amelyek szükségesek ahhoz, hogy egy AutoCAD súgófájl a Release 13 verzió súgó megjelenítője számára is elérhető legyen.

1 Nevezze át a fájlt *.ahp* kiterjesztéssel.

2 Másoljon le minden témát az azt követő két sorba is, és toldja be a #, $, és K direktíva-karaktereket. Például a `\DTPRASZTER` témát a következőképpen kell megváltoztatni:

```
\#DTPRASZTER
\$DTPRASZTER
\KDTPRASZTER
```

Ha a témának több fejezetcímkéje van, az elsőt használja a téma azonosítójaként, címeként és első kulcsszavaként, a további címkéket pedig mint kiegészítő kulcsszavakat (pontosvesszővel elválasztva).

3 A bekezdések utolsó sorait kivéve a súgószöveg minden sorát visszahajló törtjellel (\) zárja le (a direktíva szövegsorait ne). Eljárhat úgy is, hogy a súgószöveg minden bekezdésén belül megszűnteti a sor tagolását (azaz minden bekezdést egyetlen hosszú sorból álló szöveggé változtat).

4 Az utolsó téma után szúrja be a fájl vége (`\E`) direktívát.

A következő példa bemutat egy Release 12 súgófájlt, melyet ugyanezen fájlnak a fenti módszerrel Release 13 fájllá konvertált változata követ.

Release 12 súgófájl:

```
\DDOSSNAP
\ FUTÓ TÁRGYRASZTER
A DTPRASZTER parancs beállítja a futó tárgyraszter-módokat úgy, hogy
az összes következő pontkiválasztásra hatással legyen. Mialatt  a
futó raszter mód aktív, a céldoboz megjelenik a grafikus kurzor
közepén; méretét a CÉLDOBOZ parancs határozza meg.

Beállítások kiválasztása
------------------------
Annyi Tárgyraszter-ellenőrzőablakot válaszzon ki, amennyi szüséges.

Céldoboz méret
--------------
Itt beállíthatja a céldoboz méretét (maximum 20 pixel).  Ahogyan a
diát mozgatja, úgy lesz a méret az ikontételben grafikusan
megjelenítve.
\DDRENAME
A DPÁTNEVEZ parancs segítségével megváltoztathatja a Blokk nevét, a
méretezési stílust, a fóliát, a vonaltípust, a szövegstílust, az
elnevezett nézetet,  a felhasználói koordináta-rendszert, vagy á
méretablak beállításokat.

Az Elnevezett objektumok listából válassza ki azt a nevet, amit meg
akar változtatni.A Tételek lista megmutatja az átnevezhető  neveket.
Az objektumnevek megváltoztatásához írja be az aktuális nevet a Régi
név: szerkesztőablakba, majd vagy jelölje ki  a nevet a Tételek
listában, vagy írja be azt a Régi nevek szerkesztőablakba.
Billentyűzze be az új nevet az Átnevez szerkesztőablakba, majd
válassza ki a Tételek  listában az Átnevez nyomógombot az objektumnév
módosításához. A változtatás beviteléhez nyomja le az OK
nyomógombot.
```

Konvertált Release 13 súgó fájl:

```
\#DTPRASZTER
\$DTPRASZTER
\KDTPRASZTER:FUTÓ TÁRGYRASZTER
A DTPRASZTER parancs beállítja a futó tárgyraszter-módokat úgy,\
hogy az összes következő pontkiválasztásra hatással legyen. Mialatt\
a futó tárgyraszter mód aktív, a céldoboz megjelenik a grafikus\
kurzor közepén; méretét a CÉLDOBOZ parancs határozza meg.

Beállítások kiválasztása\
------------------------\
Annyi Tárgyraszter-ellenőrzőablakot válaszzon ki, amennyi szüséges.

Céldoboz méret\
--------------\
Itt beállíthatja a céldoboz méretét (maximum 20 pixel). Ahogyan a\
csúszkát mozgatja, úgy lesz a méret az ikontételben grafikusan\
megjelenítve.
\#DPÁTNEVEZ
\$DPÁTNEVEZ
\KDPÁTNEVEZ
A DPÁTNEVEZ parancs segítségével megváltoztathatja a Blokk nevét, a\
méretezési stílust, a fóliát, a vonaltípust, a szövegstílust, az\
elnevezett nézetet, a felhasználói koordináta-rendszert, vagy a\
méretablak beállításokat.

Az Elnevezett objektumok listából válassza ki azt a nevet, amit meg\
akar változtatni. A Tételek lista megmutatja azokat a neveket,\
amiket át lehet nevezni. Az objektumnevek megváltoztatásához írja\
be az aktuális nevet a Régi név szerkesztőablakba, majd vagy jelölje\
ki a nevet a Tételek listában, vagy írja be azt a Régi nevek:\
szerkesztőablakba. Billentyűzze be az új nevet az Átnevez\
szerkesztőablakba, majd válassza ki a Tételek listában az Átnevez\
nyomógombot az objektumnév módosításához. A változtatás beviteléhez\
nyomja le az OK nyomógombot.
\E
```

AutoCAD üzenetfájl

A NÉVJEGY parancs egy párbeszédpanelt nyit meg a képernyőn, amely az *acad.msg* üzenetfájl szövegét tartalmazza.

Az *acad.msg* fájl egy szövegfájl, amelyet bármely olyan editorral vagy szövegszerkesztővel létrehozhat, amely lehetővé teszi, hogy a szöveget ASCII formátumban mentse el. Az *acad.msg* fájl tartalmát helyettesítheti céges (saját cégének fejlécével) fejlécekkel, vagy felhasználóknak szóló utasításokkal. Törölheti is a fájlt, ekkor nem jelenik meg üzenet.

2

VONALTÍPUSOK ÉS SRAFFOZÁSI MINTÁK

Az AutoCAD-csomagban megtalálhatók a szabvány vonaltípusok és sraffozási minták könyvtárai. Ön felhasználhatja az így kapott eredeti vonaltípusokat és sraffozási mintákat, módosíthatja azokat, vagy létrehozhatja a sajátjait is. Ezeket a műveleteket ismertetjük ebben a fejezetben.

Ha a Postscript-kitöltést használ, akkor saját mintákat is definiálhat.

(További tudnivalók a "*fill—Saját PostScript kitöltési minták definiálása" fejezetben, a 79. oldalon)

Vonaltípus-definíciós fájlok

Az AutoCAD vonaltípusait a vonaltípus-definíciós fájlok írják le. Ezeknek a fájloknak *.lin* kiterjesztése van. Az AutoCAD vonaltípusok szóközökkel elválasztott pontok és vonalak sorozata, melyek beágyazott alak- és szövegobjektumokat is tartalmazhatnak. Az alapértelmezésbeli vonaltípus-fájl az *acad.lin*. A vonaltípusok felépítésének jobb megismeréséhez ezt a fájlt ki is nyomtathatja.

Azokat a vonaltípusokat, amelyek csak pontokból, vonalakból és az őket elválasztó szóközökből állnak, egyszerű vonaltípusoknak tekintjük. Azok a vonaltípusok, amelyek ezeken kívül még beágyazott alak- és szövegobjektumokat is tartalmaznak, az összetett vonaltípusok. Bár ezt a két fajta vonaltípust az AutoCAD hasonlóan kezeli, a definíciójuk mégis jelentősen eltér egymástól.

A vonaltípus-definiciók létrehozásának és módosításának két módszere létezik. Egyrészt szövegszerkesztővel átírhatja a *.lin* fájlt, másrészt használhatja a VTÍPUS parancs Készít opcióját is. Nincs lehetőség az összetett vonaltípusok parancssorból való létrehozására vagy módosítására.

A vonaltípus fájl sok típus definícióját tartalmazhatja. A felhasználó hozzáadhatja a saját vonaltípus-definicióját az *acad.lin* fájlhoz, vagy létrehozhat egy külön vonaltípus könyvtár fájlt. A *.lin* fájlokba megjegyzések is írhatóak, mert a rendszer minden pontosvesszővel kezdődő sort figyelmen kívül hagy.

Minden vonaltípust a *.lin* fájl két sora definiál. Az első sor megadja a vonaltípus nevét és egy opcionális leíró mezőt is kínál.

```
*vonaltípus-név [,leírás]
```

Ennek a sornak csillag (*) karakterrel kell kezdődnie, ami után közvetlenül a vonaltípus nevét kell megadni. Az opcionális leíró mezőt vesszővel kell elválasztani a névtől, és az nem lehet hosszabb 47 karakternél. Az ebben levő adatokat az AutoCAD nem használja, célja csak az, hogy segítsen a vonaltípus megjelenítésében.

A második sor az aktuális mintát leíró kódot tartalmazza.

```
igazítás,mintaleír-1,mintaleír-2, ...
```

Ez a sor a sorbaigazítási kóddal kezdődik (itt csak az A írása megengedett), amit a mintaleírók vesszőkkel elválasztott felsorolása követ (ezeket szóközzel nem lehet elválasztani).

Egyszerű vonaltípusok

Önnek lehet például egy DD1 névű egyszerű vonaltípusa, az alábbi ismétlődő mintával:

- Vonal, 0.5 rajzegység hosszú
- Szóköz 0.25 rajzegység hosszú
- Pont
- Szóköz 0.25 rajzegység hosszú

A vonaltípust így lehet definiálni:

```
*DD1,____ . ____ . ____ . ____
A,.5,-.25,0,-.25
```

Itt a DD1 a vonaltípus neve, a *leíró* mező pedig a FÓLIA Ftípus ? parancssor által kiírt vonaltípus-leírás. Ebben az esetben a leírás a pont-vonal minta egyszerű megjelenítése.

A *leíró* mező opcionális, és pontok, szóközök illetve vonalak sorozatából áll, vagy tartalmazhat megjegyzést is (például "Ez a vonaltípus takart vonalakhoz való"). Ha nincs szükség leíró mezőre, ne tegyen vesszőt a vonaltípus neve után. Ha használ leírást, az ne legyen hosszabb 47 karakternél.

Az *igazítási* mező a vonalmintának különálló vonalakhoz, körökhöz és körívekhez való igazítását teszi lehetővé. Jelenleg az AutoCAD csak egy igazítási módot támogat. Ez a második sorban az **A** karakter beírásával adható meg. A VONALTÍPUS parancs Létrehoz opciójának használatánál ez automatikusan megtörténik, de ha a vonaltípus-definíciót szövegszerkesztővel írja, akkor az **A** karaktert Önnek kell beírnia. Az AutoCAD nem fogad el semmilyen más karaktert az igazítási mezőben.

Minden *minta-n* mező egy vonaltípust alkotó kisebb egység hosszát adja meg. Ha a hossz pozítív, akkor vonalhúzás történik. Negatív szám toll-felemelést (üres szegmenst) jelent. A 0 hosszértékpontot rajzol. Egy vonaltípushoz maximum 12 vonalhosszat írhat be, feltéve, hogy ezek elférnek egy 80 karakteres sorban.

Az A típusú hozzáigazítással az AutoCAD biztosítja, hogy a szaggatott vonalak és körívek vonallal kezdődjenek, és azzal érjenek véget. Például tegyük fel, hogy létrehoz egy CENTRAL nevű vonaltípust, amely megjelenít egy, rendszerint középvonalként használt pont-vonal mintát. Ekkor az AutoCAD a pont-vonal mintát úgy alakítja át, hogy az ilyen vonaltípussal rajzolt egyenesek végpontjaira a minta vonalszakaszai essenek. A minta úgy tölti ki az egyenest, hogy annak végpontjaihoz legalább a mintakezdő vonalszakaszok fele illeszkedjen. Ha szükséges, az első és utolsó vonalszakasz meghosszabbodik. Ha a felrajzolt vonal túl rövid ahhoz, hogy legalább egy

teljes pont-vonal minta kiférjen rá, akkor az AutoCAD a végpontok között egy folyamatos vonalat húz. Körívek rajzolásakor a vonalmintázat szintén úgy rajzolódik fel, hogy a körív végpontjaira vonalszakasz essen. A köröknek nincs végpontjuk, de az AutoCAD a pont-vonal mintát úgy rajzolja fel, hogy a megjelenítés ésszerű legyen.

Az `A` típusú igazítás megköveteli, hogy az első vonalszakasz hossza 0 vagy nagyobb legyen (pont vagy meghúzott vonalszakasz). A második vonalszakasz-hossznak 0-nál kisebbnek kell lennie (szóköz). Ezen igazítási mintához legalább két vonalszakasz megadása szükséges. A kezdő és lezáró vonalszakaszok között a megadott minta ismétlődik; a másodikként megadott szakasszal végződik és az első szakasszal indítja újra a mintát, ahányszor csak szükséges.

Megjegyzés A létrehozott vonaltípus nem kerül be automatikusan a rajzba. A betöltéshez a VTÍPUS parancs Betölt opcióját kell használni.

Összetett vonaltípusok

Az összetett vonaltípusok definíciói a *.lin* kiterjesztésű fájlokba kerülnek, az egyszerű típusok definícióival együtt. Az összetett vonaltípus egy önálló, esetleg tört, szimbólumokkal kibővített vonalat ad meg. Ez a vonaltípus jelölhet segédvonalakat, határoló vonalakat, kontúrokat, stb. Az egyszerű típusúakhoz hasonlóan az összetett típusú vonalak is dinamikusan rajzolódnak fel, amikor a felhasználó megadja a csúcspontokat. A vonalba beágyazott alakok és szövegobjektumok mindig teljes egészükben kirajzolódnak, azokat soha nem vágja el a program.

Az összetett vonaltípus definíció leírási módja, az egyszerűekhez hasonlóan, mintaleírók vesszőkkel elválasztott felsorolása. Az összetett vonaltípusok mintaleíróként alakokat és szöveges objektumokat is tartalmazhatnak az egyszerű típusok pont-vonal mintái mellett.

A vonaltípus- definíció alak- és szövegobjektum-leíróinak megadási módja a következő:

alak `[alaknév,shxfájlnév]` vagy
`[alaknév,shxfájlnév,transzformálás]`

szöveg `["karakterlánc",stílusnév]` vagy
`["karakterlánc",stílusnév,transzformálás]`

ahol a *transzformálás* opcionális és az alábbiak sorozatából állhat (mindegyik előtt vessző áll)

`R=##` Relatív forgatás

`A=##` Abszolút forgatás

`S=##` Méretezés

`X=##` *X* eltolás

`Y=##` *Y* eltolás

Ebben a leírásban a # # egy előjeles decimális számot jelent (1,-17, 0.001...stb), a forgatás fokokban, a további opciók pedig vonaltípus szerinti rajzegységekben értendők. A sor eleji transzformációs betűk használata esetén azok után egyenlőségjelet és egy számot kell írni.

Alakok az összetett vonaltípusokban

Az alábbi vonaltípus-definíció egy CON1LINE nevű vonaltípust ad meg, amely egy vonalszakasz, egy szóköz és egy CON1 nevű, az *es.shx* fájlból származó beágyazott alak által alkotott ismétlődő mintából áll.

```
*CON1LINE, --- [CON1]--- [CON1]----
A,1.0,-0.25,[CON1,es.shx],-1.0
```

A szögletes zárójelekben levő kódon kívül minden más megfelel egy egyszerű vonaltípus definíciójának. Ez a példa egy olyan minimális vonaltípus-definíciót mutat be, amely egy alakot ágyaz be a vonaltípusba.

A fentebb leírt módon, hat mező együttese adhat meg egy vonaltípus részét alkotó alakot. Az első kettő kötelező és helyfüggő, a többi négy opcionális és tetszés szerinti sorrendben írható. A következő két példa; bemutatja a különböző alakdefiníciós mezők felépítését.

```
[CAP,es.shx,S=2,R=10,X=0.5]
```

Ez a kód veszi figyelembe a beillesztés előtt az *es.shx* fájlban megadott `CAP` nevű alakot a vonaltípus egységhez képest kétszeres méretben, az óramutató járása szerinti 10 fokos tangenciális elforgatással és a számolás előtti X irányú, 0.5 rajzegységgel való eltolással.

```
[DIP8,pc.shx,X=0.5,Y=1,R=0,S=1]
```

Ez a kód módosítja pc.shx alakfájlban definiált DIP8 nevű alakot úgy, hogy a beillesztés előtt X irányban 0.5 rajzegységgel, illetve Y irányban a vonaltípus fölé egy rajzegységgel eltolja.

Az összetett vonaltípusba foglalt alak definícióját az alábbi módon kell megadni:

```
[alaknév,alakfájlnév,méretezés,forgat,xeltolás,yeltolás]
```

A mezők jelentése:

alaknév A létrehozandó alak neve. Ezt a mezőt nem lehet elhagyni, mert nélküle a definíció nem érvényes. Ha az alaknév nincs benne a megadott alakfájlban, akkor a vonaltípus létrehozása a beágyazott alak nélkül folytatódik.

alakfájl-név A lefordított alakdefiníciós fájl neve (*.shx*). A mező nem hagyható ki, mert nélküle a definíció nem érvényes. Ha az *alakfájlnév* elérési útja nincs megadva,

akkor a fájlt a könyvtár útvonalon keresse. Ha meg van adva elérési út, de ott nem található a fájl , akkor vágja le az elérési út definíciót, és nézze át ismét a könyvtár útvonalat. Ha így sem található meg, akkor folytassa a vonaltípus létrehozását az alak beágyazása nélkül.

nagyítás `S=érték`. Az alak léptéke olyan, méretezési faktorként használatos, amellyel az alak belsőleg definiált mérete megszorzódik. Ha a belsőleg definiált méret 0, akkor az *S=érték* önmagában fejezi ki a méretet.

forgat `R=érték` vagy `A=érték`. Az R= kifejezi a relatív vagy érintőleges forgatást a vonal létrehozásához képest. Az A= megadja az alak abszolút forgatását a kezdőponthoz képest. Minden alak elfordulása megegyezik, függetlenül a vonalhoz képest elfoglalt relatív helyzetüktől. A kifejezés kiegészíthető a d karakterrel, a fokok kifejezésére (ha nincs beírva, ez az alapértelmezés); r-rel (radiánt), és a g-vel (grad). Ha nincs megadva forgatási érték, az 0 relatív forgatást jelent.

xeltolás `X=érték`. Az X eltolás megadja az alak eltolódását a vonaltípus definíciójának csúcspontjából számított vonaltípus X tengelye mentén. Ha az X eltolás nincs megadva, vagy értéke 0, az alak eltolódás nélkül kerül kialakításra. Használja ezt a kifejezést, ha folytonos vonalat kíván szerkeszteni az alakokkal. Ez az érték nem szorzódik az S= által megadott méretezési faktorral.

yeltolás `Y=érték`. Az Y eltolás megadja az alak eltolódását a vonaltípus definíciójának csúcspontjából számított vonaltípus Y tengelye mentén. Ha az Y eltolás nincs megadva, vagy értéke 0, az alak eltolódás nélkül kerül kialakításra. Ez az érték nem szorzódik az S= által megadott méretezési faktorral.

Szöveg az összetett vonaltípusokban

A szöveges összetett vonaltípusokba szövegrészletek ágyazhatóak be, az alakzatokhoz hasonlóan. A legfontosabb különbség a szöveg és az alakok használata között az, hogy a szöveg rajzbeli szövegstílushoz kapcsolódik, míg az alakokat egy alakleíró fájl közvetlenül határozza meg. A vonaltípushoz kapcsolódó stílusnak már a vonaltípus rajzba töltése előtt léteznie kell.

Íme egy példa a szöveget tartalmazó összetett vonaltípusra.

```
 *MCline, --- MC --- MC ---
A,1.0,-0.25,["MC",mystyle,S=1,R=0,X=0,Y=-0.25],-1.25
```

Ahol `mcline` a vonaltípus neve és `--- MC --- MC ---` az ASCII leírása. A vonaltípus definíció második sorának megadási módja:

`["karakterlánc",stílus,S=nagyítás,R=forgatás,X=xeltolás, Y=yeltolás]`

A megadási módban előforduló mezők definíciói.

karakterlánc Az összetett vonaltípusba bekerülő szöveg.

stílus A kidolgozandó szövegstílus neve. A megadott szövegstílusnak léteznie kell. Ha megadása elmarad, a rendszer az aktuális stílust használja.

méretezés S=érték. A stílus léptéke olyan méretezési faktorként használatos, amellyel a szöveg magassága megszorzódik. Ha a szövegmagasság 0, az S=érték önmagában adja a méretet.

forgatás `R=érték` vagy `A=érték`. Az R= kifejezi a relatív vagy érintőleges forgatást a vonal létrehozásához képest. Az A= megadja az alak abszolút forgatását a kezdőponthoz képest. Minden alak elfordulása megegyezik, függetlenül a vonalhoz képest elfoglalt relatív helyzetüktől. A kifejezés kiegészíthető a d karakterrel, a fokok kifejezésére (ha nincs beírva, ez az alapértelmezés); r-rel (radián), és a g-vel (grad). Ha nincs megadva forgatási érték, az 0 relatív forgatást jelent.

A forgatás középpontja az alapvonal és a névleges betűmagasságot jelző doboz között helyezkedik el.

xeltolás `X=érték`. Az X eltolás megadja az alak eltolódását a vonaltípus definíciójának csúcspontjából számított vonaltípus X tengelye mentén. Ha az X eltolás nincs megadva, vagy értéke 0, az alak eltolódás nélkül kerül kialakításra. Használja ezt a kifejezést, ha folytonos vonalat kíván szerkeszteni alakokkal. Ez az érték nem szorzódik az S= által megadott méretezési faktorral.

yeltolás `Y=érték`. Az Y eltolás megadja az alak eltolódását a vonaltípus definíciójának csúcspontjából számított vonaltípus Y tengelye mentén. Ha az Y eltolás nincs megadva, vagy értéke 0, az alak eltolódás nélkül kerül kialakításra. Ez az érték nem szorzódik az S= által megadott méretezési faktorral.

Sraffozási minták készítése

Az AutoCAD sraffozási minták elkészítéséhez egy kevés ismeretre, némi gyakorlatra és türelemre, valamint egy szövegszerkesztőre van szükség. A sraffozási mintát beillesztheti az *acad.pat* könyvtárfájlba, vagy elmentheti egy önálló fájlba is, de ez utóbbi esetben a fájl nevének meg kell egyeznie a minta nevével (például a "pit" nevű mintának a *pit.pat* nevű fájlba kell kerülnie).

A sraffozási minta formátuma — a tárolási helyétől függetlenül — mindig azonos. Tartalmaz egy alábbihoz hasonló fejsort:

```
*minta-neve [, leírás]
```

és egy vagy több vonalmeghatározást az alábbi formában:

```
szög, x-origó, y-origó, delta-x, delta-y [, vonal-1, vonal-2, ...]
```

Az üres sorokat és a pontosvesszőtől jobbra lévő szövegeket az AutoCAD nem veszi figyelembe.

Egy V45 nevű minta például, amely 45 fokban döntött, egymástól 0.5 egység távolságra lévő vonalakkal sraffoz, a következőképpen definiálható:

```
*L45,45 fokos ferde vonalak
45, 0,0, 0,0.5
```

Ez az egyszerű definíció meghatározza, hogy a vonalak 45 fokban dőljenek, a sraffozó vonalak csoportjának első vonala a (0,0) origón haladjon át, és az egy csoportba tartozó sraffozó vonalak közötti távolság 0.5 rajzegység legyen. A minta neve V45 választható ki, a *leírás* mező pedig a mintának azt az azonosítóját tartalmazza, amely a SRAFFOZ ? parancs hatására fog megjelenni. (Ezt a leírást el is hagyhatja, de ebben az esetben a minta nevét nem követheti vessző) A mintaleíró fájl minden sora maximum 80 karaktert tartalmazhat.

Az AutoCAD mintafájlja—*acad.pat*

Az AutoCAD által használt alapértelmezés szerinti sraffozási minták az *acad.pat* fájlban vannak. Az előbbiekben leírtak szerint hozzáadhat mintadefiníciókat ehhez a fájlhoz, vagy létrehozhatja saját fájljait.

A HSRAFFOZ parancs Válasszon sraffozási mintát párbeszédpaneljében az *acad.pat* fájlban definiált összes sraffozási minta neve megjelenik. Amennyiben új sraffozási mintával kívánja kiegészíteni a párbeszédpanelt, helyezze el annak definícióját az acad.pat fájlban.

A HSRAFFOZ parancs párbeszédpaneljében az *acad.slb* diafájlban található mindazon diák megjelennek, amelyeknek neve megegyezik az *acad.pat* mintáinak valamelyikével.

A sraffozott minták felépítése

A minta egy vagy több *mintavonalból* áll (a mintában szereplő vonalak számának nincs felső határa). Minden mintavonalat úgy tekintünk, mint egy *vonalcsoport* első tagját. A csoportot az AutoCAD úgy képezi, hogy az első vonallal — delta elmozdulásértékkel mindkét irányban eltolva — végtelen sok párhuzamos vonalat hoz létre. A *delta-y* érték a csoport egyes tagjai közötti távolságot jelöli (azaz a vonalakra merőlegesen mért távolságot). A *delta-x* a csoport tagjainak a vonalak irányában mért eltolását jelöli — ennek csak szaggatott vonalak esetében van jelentősége. (A vonalakat végtelen hosszúságúnak kell tekinteni; a szaggatott vonal erre kerül rá.)

A sraffozás művelete abból áll, hogy a minta definíciójában szereplő összes mintavonalat párhuzamos vonalak végtelen halmazává bővítjük. Az AutoCAD minden kiválasztott rajzelemet megvizsgál, vajon metszi-e ezeket a vonalakat. Minden egyes metszéspont a sraffozó vonalat — a kiválasztott sraffozási stílustól függően — ki- vagy bekapcsolja. Ha a sraffozó vonal szaggatott, akkor azokon a területeken, ahol a sraffozás be van kapcsolva, a vonalakat az AutoCAD szaggatott mintával jeleníti meg.

A szomszédos területek pontos illeszkedését az biztosítja, hogy a sraffozó vonalak minden csoportja egy abszolút origóval rendelkező kiinduló vonalból jön létre párhuzamos eltolásokkal.

Minták szaggatott vonalakból

Szaggatott mintavonalat úgy definiálhat, hogy a vonalmeghatározás végéhez vonalhosszúság tételeket fűz hozzá. Minden tétel a vonalat alkotó egy-egy szakasz hosszát határozza meg. Ha értéke pozitív, akkor ez egy leengedett tollal ténylegesen megrajzolt szakasz lesz. Ha a hosszúság értéke negatív (felemelt toll), akkor ezt a vonalszakaszt az AutoCAD nem rajzolja ki. A minta az origóban kezdődik az első szakasz kirajzolásával, és a megadott szakaszokkal ciklikusan ismétlődik. A nulla hosszúságú szakasz egy pont rajzolásának felel meg. Mintavonalanként legfeljebb 6 vonalszakasz határozható meg.

Például módosítsuk a 45 fokos dőlésszögű sraffozási mintát úgy, hogy 0.5 egység hosszúságú szakaszokból álló szaggatott vonalakat rajzoljon, és a vonalszakaszok közötti üres szakaszok hossza is 0.5 egység legyen. Ez a minta a következőképpen definiálható:

```
*SZAGGAT45,45 fokos ferde szaggatott vonal
45, 0,0, 0,.5, .5,-.5
```

Ez megegyezik az eredeti 45 fokos dőlésű mintával, csak a végéhez hozzáírtuk a vonalszakaszok meghatározását. A kitűzött célnak megfelelően a leengedett tollal megrajzolt szakasz hossza 0.5, és a felemelt tollal kihagyott szakasz hossza is 0.5. Tételezzük fel, hogy egy 0.5 egység hosszúságú szakaszt, egy 0.25 egység hosszúságú üres részt,

egy pontot, majd a következő vonalszakasz előtt ismét egy 0.25 egység hosszúságú üres részt szeretnénk kirajzoltatni. Ez az alábbi definícióval valósítható meg:

```
*PVONAL45,45 fokos vonal-pont-vonal minta
45, 0,0, 0,.5, .5,-.25,0,-.25
```

Most vizsgáljuk meg a *delta-x* paraméter hatását a szaggatott vonalakból álló csoportra. Először tekintse meg a következő definíciót:

```
*AZONOS
0, 0,0, 0,.5, .5,-.5
```

Ez egymástól .5 egységnyi távolságra lévő vonalak csoportját rajzolja meg, amelyek mindegyike egyenlő hosszúságú vonalakra és üres részekre van felosztva. Mivel a *delta-x* értéke nulla, a vonalszakaszok azonosan helyezkednek el a csoport minden vonalán belül. Egy, ezzel a mintával sraffozott terület az alábbi képet mutatná:

```
---------
---------
---------
```

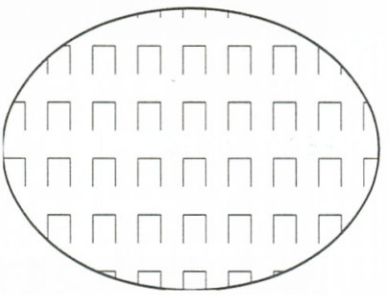
Most változtassuk meg a mintát az alábbiak szerint:

```
*ELTOLT
0, 0,0, .5,.5, .5,-.5
```

Ez csak annyiban tér el az előzőtől, hogy a *delta-x* értékét ezúttal 0.5 egységre állítottuk be. Ezzel a csoport egymás után következő tagjainak mindegyikét a vonal irányában (esetünkben az *X* tengellyel párhuzamosan) 0.5 egységgel elmozdítottuk. Mivel a vonalak végtelen hosszúak, ezért ennek olyan a hatása, mintha a szaggatott minta csúszna le a megadott értékkel. Ebben az esetben a sraffozott területen az alábbi képet kapnánk:

```
---------
---------
---------
```

Az eddig bemutatott minták origója a (0,0) pont volt, ennélfogva a vonalcsoport egyik tagja — vonalszakasszal kezdődően — ebből a pontból indult ki. Az összetettebb minták készítésénél szükség van arra, hogy minden egyes vonalcsoporthoz körültekintően válasszunk kezdőpontot, eltolási értéket és mintát. Tételezzük fel, hogy olyan mintát szeretnénk készíteni, amely felfordított U betűkből áll. Vagyis egy vonalat felfelé, egyet vízszintesen, egyet pedig lefelé kell szerkesztenünk; más szóval egy olyan négyzetet, melynek az alsó oldala hiányzik. Ebből olyan mintát definiáljunk, amely 1 rajzegységenként ismétlődik. Mindegyik minta 0.5 egység magas és széles legyen, mint azt az alábbi ábra illusztrálja.



Ezt a mintát a következőképpen definiálhatjuk:

```
*FORDU, Megfordított U betűk
90, 0,0,  0,1, .5,-.5
0, 0,.5, 0,1, .5,-.5
270, .5,.5 0,1, .5,-.5
```

Az első vonal (felfelé tartó szakasz) egy egyszerű szaggatott vonal, amelynek origója a (0,0) pontban van. A második vonalnak (felső szakasz) a felfelé tartó szakasz végpontjában kell kezdődnie, ezért az origóját a (0,.5) pontba kell állítanunk. A harmadik vonalnak (lefelé tartó szakasz) a felső szakasz végpontjában — a minta első elemének kirajzolásakor a (.5,.5) pontban — kell kezdődnie, így az origóját ebbe a pontba kell állítanunk. A minta harmadik vonalát az alábbiak szerint is megadhattuk volna:

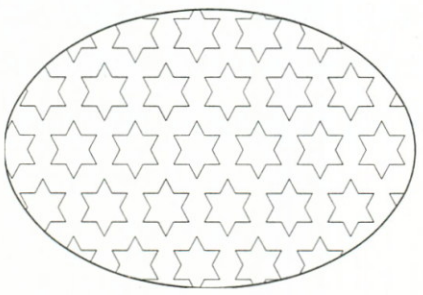
```
90, .5,0, 0,1, .5,-.5
```

vagy

```
270, .5,1, 0,1, -.5,.5
```

A szaggatott vonal mintája az origóból indul ki, és a dőlésszöggel meghatározott vektor irányában halad. Ily módon két, egymáshoz képest 180 fokban elfordított szaggatott vonalcsoport nem ugyanazt a mintát eredményezi. Folytonos vonalak esetében azonban természetesen igen.

Tekintse meg az alábbi, hatágú csillagokból álló mintát:



Ez a példa segítségére lehet abban, hogy fejlessze a minták definíciójával kapcsolatos ismereteit. (Segítségül: a 0.866 érték a 60 fok szinusza.) Íme a minta AutoCAD definíciója:

```
*CSILLAGOK, Dávid-csillag
0, 0,0, 0,.866, .5.-.5
60, 0,0, 0,.866, .5.-.5
120, .25,.433, 0,.866, .5,-.5
```

3

ALAKOK, FONTOK ÉS A POSTSCRIPT TÁMOGATÁSA

Az alakok olyan objektumok, melyeket a blokkokhoz hasonlóan használhat. Ezek használatához először töltse be a BETÖLT parancs segítségével az alakok definícióját tartalmazó, már lefordított alakfájlt. Azután adja ki az ALAK parancsot, amely a fájlból a rajzba illeszti az alakot. Minden egyes alakra megadhatja, hogy az milyen léptékkel és forgatással kerüljön a rajzba. A szövegfontokat és az alakokat azonos módon definiáljuk.

A blokkok sokkal könnyebben és szélesebb körben használhatóak, mint az alakok. Ez utóbbiakat azonban az AutoCAD hatékonyabban tárolja és rajzolja ki. A felhasználó által definiált alakok gyorsíthatják a munkát, ha egy részletre többször is szükség van, és számít az idő.

Ez a fejezet bemutatja, hogyan hozzon létre egy saját alak- és fontfájlt rajzszimbólumok és szövegfontok számára, valamint ismerteti az AutoCAD továbbfejlesztett PostScript szolgáltatásait is.

Az alak- és fontfájlok fordítása

Az alakokat egy speciális formátumú szövegfájl írja le, amelynek kiterjesztése *.shp*. Ilyen fájlok készítéséhez olyan szövegszerkesztőt használjon, amely alkalmas ASCII formátumú fájlok elmentésére. Ezt az ASCII fájlt kell azután lefordítania. Az alakdefiniáló fájlokból fordításkor a BETÖLT és a STÍLUS parancsok által elfogadott formátumú fájl képződik.

Egy alak- vagy fontfájl fordításához billentyűzze be:

Parancs: **fordít**

Az AutoCAD "Válasszon alakfájlt" címmel egy párbeszédpanelt jelenít meg, amelyből kiválaszthat egy alakfájlt (*.shp*) vagy egy PostScript fontfájlt (*.pfb*). A fájlnév kiválasztása után a fordítási folyamat megkezdődik. Ha az AutoCAD az alak leírásában hibával találkozik, akkor üzenetet jelenít meg, közölve a hiba típusát és annak a sornak a számát, amelyben a hibát felfedezte. A fordítás befejeztével a következő üzenet jelenik meg:

Sikeres fordítás.
A(z) *név.shx* kimeneti fájl *nnn* bájtot tartalmaz.

A lefordított fájl neve megegyezik az alak definiáló fájléval, de kiterjesztése *.shx*. Ha az alakdefiniáló fájl egy fontdefiníciót tartalmaz, használja a STÍLUS parancsot egy szövegstílus definiálására. Ezután válassza ki valamelyik szövegáthelyező parancsot (például a DSZÖVEG vagy MSZÖVEG) a karakterek szövegbe illesztéséhez. Ha az alakdefiniáló-fájl alakdefiníciót tartalmaz , használja a BETÖLT parancsot, hogy az alakot a rajzba tölthesse. Végül az ALAK parancs segítségével az egyedülálló alakokat is beillesztheti a rajzba (hasonlóan a BEILL parancshoz).

PostScript Fontok fordítása

Az AutoCAD a legtöbb Type 1 PostScript fontot képes betölteni vagy lefordítani. Ez a fontformátum ipari szabvánnyá vált a fontok kódolásában és forgalmazásában. Az Adobe Type 1 fontformátum részletes leírását az *Adobe Type 1 Font Format Version 1.1* kézikönyvben találja meg. Miután megvásárolta és telepítette ezeket a fontokat, elkezdheti használni őket az AutoCAD programmal.

A Type 1 fontok dinamikus betöltéséhez (ami a STÍLUS parancs segítségével meghatározott *.pbf* fájlokkal hajtható végre) szükség van a fontspecifikáció elemzésére. Ez a PostScript nyelv egy speciális célú változatában íródott, és az AutoCAD szöveglétrehozó parancsainak megfelelő készletet generál. Ez körülbelül annyi ideig tart, amennyit az AutoCAD a saját alak- és fontfájlainak (*.shp*) bináris formává alakításával tölt el. Ha egy rajz Type 1 fontokra hivatkozik, akkor az

AutoCAD normális esetben az összes fontot lefordítja a rajz betöltésekor; ez is növelheti a rajz betöltésének idejét.

A késlekedést elkerülheti a gyakran használt fontok PostScript fordításával (*.shx*); ezeket az AutoCAD sokkal gyorsabban tölti. Ha egy AutoCAD szövegelem *.shx* fájlformában is elérhető fontnévre hivatkozik, akkor az AutoCAD ezt használja az eredeti PostScript font helyett.

A Postscript fontok lefordított fájljai igen nagy helyet foglalnak a lemezen, így csak azokat a fontokat fordítsa le, amelyeket gyakran használ.

Megjegyzés Ellenõrizze az Ön által használt Postscript fontok szerzői jogait. Azonos jogok vonatkoznak a lefordított *.shx* formátumra is.

Hibák a Postscript fontokban

A betöltés és a fordítás során előforduló legtöbb hiba azzal jár, hogy a font nem tölthető be. Erre az AutoCAD hibaüzenetei figyelmeztetik.

Figyelmeztetések és korlátozások

Nincs rá garancia, hogy az AutoCAD minden Type 1 fontot képes lefordítani és betölteni. Az AutoCAD PostScript szolgáltatásait csak az Adobe típusú fontok szűkebb készletének feldolgozására szántuk.

Alakdefiniáló fájlok

Az AutoCAD font- és alakfájljait (*.shx*) az alakdefiniáló fájlokból generálhatjuk (fordíthatjuk le). Az alakdefiniáló fájlt bármely olyan editorban vagy szövegszerkesztő programban elkészítheti és módosíthatja, amelyik képes fájlokat ASCII formátumban elmenteni.

Az alak leírásának szintaktikája minden egyes alaknál (vagy karakternél) ugyanaz, tekintet nélkül az arra, hogy mire fogja használni ezeket. Ha egy alakdefiniáló fájlt fontfájlként használ, akkor az első sor magát a fontot írja le, nem pedig a fájlban levő alakot. Ha az első sor alakot definiál, akkor a fájl alakfájlként használható.

Az AutoCAD lemezeken mintaként két alakfájlt talál: a *pc.shx* és az *es.shx* fájlokat. Az egyiket nyomtatott áramköri lapok tervezéséhez, a másikat elektronikus kapcsolási vázlatok készítéséhez használhatja. Ezen fájlok szöveges változatának (*pc.shp*, illetve *es.shp*) tanulmányozása, továbbá a bennük szereplő alakleírások megváltoztatása segítségére lehet az AutoCAD alakok definiálásának elsajátításában.

Alakleírás

Az alakdefiníció minden egyes sora 128 karaktert tartalmazhat. A hosszabb sorok nem fordítódnak le. Az AutoCAD figyelmen kívül

hagyja az üres sorokat és a pontosvesszőktől jobbra álló karaktereket. A pontosvessző lehetővé teszi belső megjegyzések elhelyezését az alakdefiníciós fájlon belül.

Minden alakdefinícióhoz egy, az alábbi formájú fejsor tartozik, amelyet egy vagy több sorban követnek az alakot meghatározó, egymástól vesszővel elválasztott adatbájtok, melyek végén egy 0 áll.

```
*alakszám, defbájtok, alaknév
azonbájt1,azonbájtc2,azonbájtc3,...,0
```

Az alábbiakban leírjuk az alakot definiáló mezőket:

alakszám Egy a fájlt jelölő szám1 és 255 között, melyet egy csillag (*) előz meg. Egy alakfájlon belül minden alaknak kell, hogy legyen száma (a 256, 257 és 258 Degree_Sign, Plus_Or_Minus_Sign, és Diameter_Symbol azonosítók által foglaltak). A szövegfontoknak az ASCII kódtáblában található karakterhez tartozó száma van (a fájlokban minden egyes karakterhez adott alakdefiníció tartozik). Egyéb alakok bármilyen értéket kaphatnak.

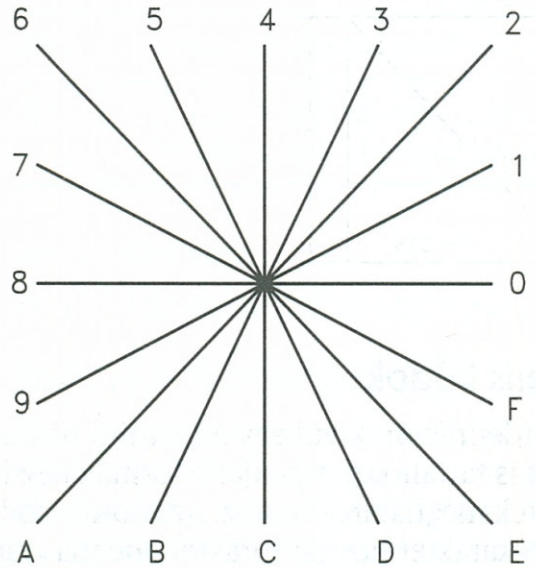
defbájtok A defbájt az alak leírásához szükséges adatbájtok (*azonbájtok*) számát jelöli, a sort lezáró nullát is beleértve. A felső határ alakonként 2000 bájt.

alaknév Az alak neve. Nagybetűvel írandó, hogy a program felismerje. A kisbetűket tartalmazó neveket a program figyelmen kívül hagyja és általában font-alak definíció megjelöléseként értelmezi.

azonbájt Alakazonosító bájt. Minden azonosító bájt egy kód, amely vagy egy vektor hosszát és irányát adja meg, vagy egy speciális kódot. Az azonosító bájtot az alak definíciós fájlban adhatja meg akár decimális, akár hexadecimális formában. Az ebben a részben található példák mindkét formát bemutatják (mint ahogyan az alakdefiníciós fájlban is előfordul mindkettő). Ha az azonosító bájt első karaktere 0, a következő két karaktert hexadecimális értékként kell figyelembe venni.

A vektorhossz- és iránykódok

Egy vektor hossza és iránya egyetlen alak azonosító bájtban (egy `azonbájt` mezőben) van megadva. Minden vektor hosszúság- és iránykódja egy három karakterből álló karakterlánc. Az első karakternek 0 értékűnek kel lennie, ez jelzi az AutoCAD programnak, hogy a következő két karakter hexadecimális lesz. A második karakter a vektor hosszát adja meg az adott egységben. Az érvényes hexadecimális tartomány 1 (egy egység hosszú) és F (15 egység hosszú) között van. A harmadik karakter a vektor irányát adja meg. A kódolást a következő ábra mutatja be.



Vektorirány-kódok

Az ábrán látható valamennyi vektorhoz ugyanaz a hosszmeghatározás tartozik. Az átlós vektorok megnyúltak annyira, hogy megfeleljenek a legközelebbi ortogonális vektor X, illetve Y elmozdulásértékének. Ez hasonló az AutoCAD raszterháló alkalmazásához.

A következő példa létrehoz egy DOBOZ nevű alakot, amelynek alakszáma legyen egy tetszőlegesen kiválasztott szám, például 230.

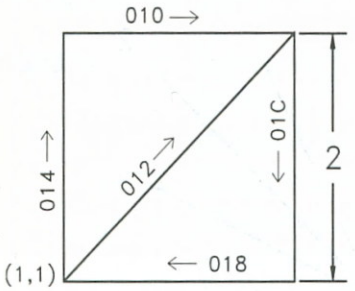
```
*230,6,DOBOZ
014,010,01C,018,012,0
```

A fenti bájtsorozat egy olyan négyzetet ad meg, amelynek magassága és szélessége egységnyi, és amelyben egy átlós vonal fut a bal alsó sarokpontból a jobb felső sarokpontba. Miután *ddoboz.shp* néven elmentette a fájlt, használja a FORDÍT parancsot a *ddoboz.shx* fájl generálásához. Mentse el a BETÖLT parancs segítségével a definíciót

tartalmazó alakot, majd használja az ALAK parancsot a következőképpen:

Parancs: **alak**
Alaknév (vagy ?): **doboz**
Kezdőpont: **1,1**
Magasság <*alapértelmezés*>: **2**
Forgásszög <*alapértelmezés*>: **0**

Ennek eredményeképpen a következő alak jelenik meg:



Speciális kódok

A vektordefiníción kívül egy azonosító bájt a következő speciális kódokat is tartalmazhatja újabb formák készítéséhez és bizonyos műveletek meghatározásához. Azonosító kódot úgy használhat, hogy a három karakter hosszú karakterlánc második karakterének (a vektorhosszúság azonosítójának) 0-t ad meg. Megteheti azt is, hogy csak a kód számát adja meg. (Például mind a 008, mind a 8 érvényes azonosítók).

Azonosító bájt kódok

Kód	*Leírás*
000	Az alakdefiníció vége
001	Bekapcsolja a rajzmódot (toll le)
002	Kikapcsolja a rajzmódot (toll fel)
003	A vektorhosszakat elosztja a következő bájt értékével
004	A vektorhosszakat megszorozza a következő bájt értékével
005	Az aktuális vektorhosszat a veremtárolóba helyezi (stack)
006	Az aktuális pozíciót kiemeli a veremtárolóból (stack)
007	A következő bájttal megadott alárendelt alakot rajzolja meg

Azonosító bájt kódok (*folytatás*)

Kód	*Leírás*
008	A következő két bájt által megadott X-Y elmozdulás
009	Többszörös X-Y elmozdulás, (0,0)-val lezárva
00A	A következő két bájt egy nyolcad ívet határoz meg
00B	A következő öt bájt egy ív törtrészét határozza meg
00C	X-Y elmozdulással és kihajlással meghatározott ív
00D	Több, kihajlással meghatározott ív
00E	A következő parancsot csak függőleges stílusú szöveg esetében kell figyelembe venni

Kód 0: Alak vége

Ez a kód az alakdefiníció végét jelöli.

Kód 1 és 2: Rajzmód vezérlése

Ezek a kódok a Rajzmódot vezérlik. A Rajzmód az alak rajzolásának kezdetekor mindig be van kapcsolva. A Rajzmód bekapcsolt állapotában a vektorok kirajzolják a vonalakat, kikapcsolt állapotában viszont a vektorok rajzolása nélkül mozdul el a rajztoll a következő pozícióba.

Kód és 3 és 4: Méretbeállítás

A 3-as és 4-es kódok az egyes vektorok relatív méretét szabályozzák. Az ALAK parancs promptjára megadott magasságot az AutoCAD kiindulásként a 0, 4, 8 vagy C irányú egyszerű ortogonális vektorok hosszaként értelmezi. A 3-as vagy 4-es kódot egy egész értékű (1-től 255-ig) méretarány-tényezőt tartalmazó bájt követi. Amennyiben azt kívánja, hogy a megadott magasság az egész alak méretére vonatkozzon, de az 10 vektorhossznak feleljen meg, akkor a 3,10 értékeket kell megadnia a magassági méret léptékének a beállításához. Egy alakon belül a méretarány-tényezők halmozódnak, azaz egy 2-es szorzó után egy 6-os szorzó 12-es méretarány-tényezőt eredményez. Az alak végén a méretarány-tényező eredeti értékét általában vissza kell állítani, különösen alárendelt alakok és szövegfontok esetében. Az AutoCAD ezt automatikusan nem teszi meg.

Kód 5 és 6: Pozíció elmentése és visszaállítása

Az 5-ös és 6-os kódok egy alak rajzolása közben elmentik (push), illetve visszaállítják (pop) az aktuális koordinátákat, így ebbe a pozícióba az alak rajzolása során később visszatérhet. Minden elmentett pozíciót vissza is kell állítania! A pozíciók veremtárolója (stack) csak 4 helyzetet

képes tárolni. A verem túlcsordulását (túl sok push, illetve hiányzó pop-ok miatt) a következő üzenet jelzi az alak kirajzolásakor.

Pozíciós verem túlcsordulása a(z) nnn alakban

Hasonlóképpen, ha több pozíciót próbál visszaállítani, mint amennyit elmentett a veremtárolóban, akkor az alak kirajzolásakor az alábbi üzenet jelenik meg:

Pozíciós verem alulcsordulása a(z) nnn alakban

Kód 7: Alárendelt alak

A 7-es kód egy alárendelt alakra történő hivatkozást jelöl. A 7-es kódot követő azonosító bájt egy alakszám 1 és 255 között. Az ezzel a számmal jelölt alakot (amelynek ugyanabban az alakfájlban kell lennie) fogja az AutoCAD kirajzolni. Fontos megjegyezni, hogy az új alak számára a Rajzmódot az AutoCAD nem állítja alaphelyzetbe. Miután az alárendelt alak kirajzolása befejeződött, az aktuális alak rajzolása folytatódik.

Kód 8 és 9: X-Y elmozdítások

A normál vektorazonosító bájtok csak a 16 előre meghatározott irányban képesek rajzolni, és a maximális hosszuk is csak 15 lehet. Ezek a megszorítások az alakdefiníciókat igen hatékonnyá teszik, de sok esetben korlátot is jelentenek. A 8-as és 9-es kódok azért állnak rendelkezésre, hogy az X-Y elmozdulások segítségével a nem szabványos vektorok kirajzolását is lehetővé tegyék. A 8-as kódot két bájtnak kell követnie az alábbi formában:

```
8,X-elmozdulás,Y-elmozdulás
```

Az X és Y elmozdulások értékének -128 és +127 közé kell esnie. A + jel használata opcionális, sőt zárójeleket is használhat az olvashatóság javítása érdekében. Például a

```
8, (-10,3)
```

egy olyan vektort eredményez, amely 10 egységnyit balra, 3 egységnyit felfelé mozdul (rajzol). A két elmozdulásbájt után az alak visszatér a normál vektormódba.

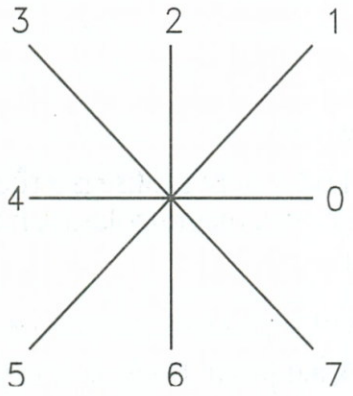
A 9-es kódot akkor használhatja, ha nem szabványos vektorok sorozatát szeretné kirajzolni. Ezt a kódot tetszőleges számú X-Y elmozduláspár követheti, egy (0,0) párossal lezárva. Például a

```
9, (3,1), (3,2), (2,-3), (0,0)
```

bájtsorozat három nem szabványos vektort rajzol, majd visszatér a normál vektormódba. Fontos megjegyeznie, hogy az X-Y elmozduláspárok sorozatát egy (0,0) párral kell lezárni annak érdekében, hogy az AutoCAD az ezt követő normál vektorokat vagy speciális kódokat felismerje.

Kód 00A: Nyolcadív

A 00A (vagy 10) speciális kódot követő két bájt egy ív megadására szolgál. Az ilyen típusú ívet nyolcadnak vagy nyolcadívnek nevezzük, mivel a teljes kör egy vagy több 45 fokos nyolcadára terjed ki, és mindig egy-egy nyolcadhatáron kezdődik és végződik. A nyolcadhatárokat a "3 óra" pozíciójából kiindulva, az óramutató járásával ellentétes irányban számozzuk., mint azt az alábbi ábra mutatja:



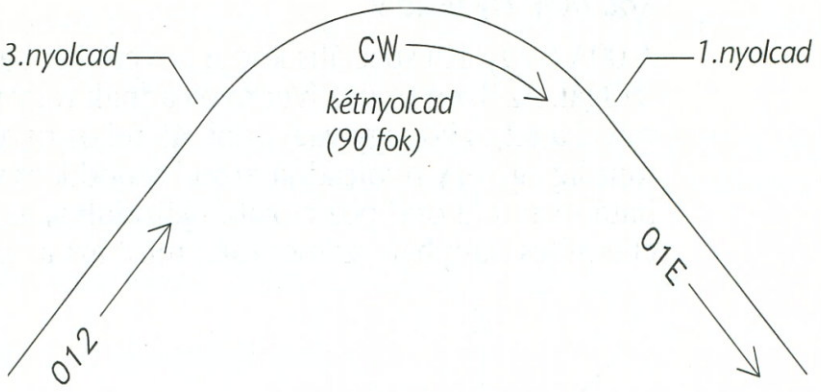
Az ív meghatározása:

```
10,sugár,(-)0SC
```

A sugár értéke tetszőleges egész szám lehet 1-től 255-ig. A második bájt jelöli az ív irányát (pozitív érték esetén az óramutató járásával ellentétes, negatív érték esetén azzal megegyező), a kiinduló nyolcadhatár számát (S, értéke 0-tól 7-ig terjedhet), valamint a nyolcadokban kifejezett kiterjedését (C, értéke 0-tól 7-ig terjedhet, ahol a 0 nyolc nyolcadot, azaz egy teljes kört jelent). Az olvasás megkönnyítésére zárójelek is használhatók. Példaként tekintse meg egy alakdefiníció alábbi részletét:

```
...012,10,(1,-032),01E,...
```

Ez egy egységnyi vektort rajzol felfelé és jobbra, egy egységnyi sugarú ívet az óramutató járásával egyező irányban a 3. nyolcadhatártól kezdve 2 nyolcadon keresztül, majd egy egységnyi vektort jobbra lefelé, mint a következő ábrán látható.



Kód 00B: Ív törtrésze

A 00B (11) speciális kód a nem szükségszerűen egy nyolcadhatárnál kezdődő vagy végződő ívek megrajzolására használható. A definíció öt bájtot foglal magába:

```
11,kezdőérték,végérték,felső-sugár,alsó-sugár,(-)0SC
```

A kezdő- és a végérték azt jelöli, hogy az ív egy nyolcadhatártól milyen messze kezdődik, illetve végződik. A <felső sugár> a sugár magasabb helyiértékű 8 bitjét jelöli; amennyiben a sugár nem nagyobb 255 egységnél, ennek értéke nulla. Egy 255-nél nagyobb értékű ívsugár létrehozásához, szorozza meg a `felső-sugár` értékét 256-tal és adja hozzá azt a számot, ami a `sugár` értékéhez tartozik. A sugárra és a lezáró azonosító bájtra ugyanaz vonatkozik, mint a nyocadív azonosítójára (kód 00A az előzőekben leírtak).

A kezdőérték meghatározásához ki kell számítania a kezdő nyolcadhatár (a 45 fok többszöröse) és az ív kezdetének különbségét fokokban. Ezt a különbséget szorozza meg 256-tal, majd ossza el 45-tel. Ha az ív egy nyolcadhatáron kezdődik, akkor a kezdőértéke nulla.

A végértéket a kezdőértékhez hasonló módon számíthatja ki, azzal az eltéréssel, hogy az ív vége és az utolsó átlépett nyolcadhatár közötti szögkülönbségből kell kiindulnia. Ha az ív egy nyolcadhatáron végződik, akkor a végértéke nulla.

Az a 3 egység sugarú ív például, amely az 55 foktól a 95 fokig terjed, a következőképpen kódolható:

```
11,(56,28,0,3,012)
```

Magyarázat:

```
kezdeti_eltolás   = 56   mivel ((55 - 45) * 256 / 45)  = 56
vég_eltolás       = 28   mivel ((95 - 90) * 256 / 45)  = 28
magas_sugár       = 0    mivel (sugár < 255)
sugár             = 3
kezdő nyolcad     = 1    mivel az ív a 45 fokos nyolcadban
                            kezdődik
végső nyolcad     = 2    mivel az ív a 90 fokos nyolcadban
                            ér véget
```

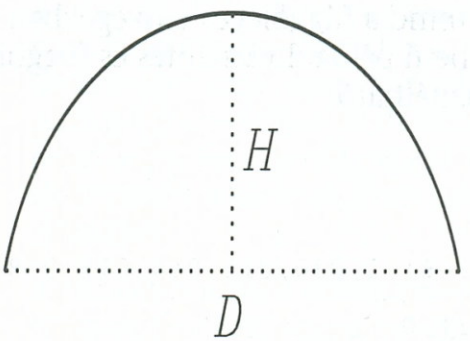
Kód 00C és 00D: Kihajlásával megadott ív

A 00C és 00D (12 és 13) speciális kódok az ívszakaszok meghatározásának egy másik lehetőségét biztosítják az alakdefiníciókban. Annyiban hasonlítanak a 8-as és 9-es kódokra, hogy szintén használhatók *X-Y* elmozdulás megadására. A 00C és 00D kódok azonban azt is lehetővé teszik, hogy az elmozdulásvektorhoz rendelt *kihajlási tényező* segítségével rajzoljon íveket. A 00C kód egyetlen ívet határoz meg, a 00D kód viszont több, egymáshoz csatlakozó ívet (*íves szakaszokból álló vonalláncot*) rajzol, amíg egy záró (0,0) elmozdulással nem találkozik.

A 00C kódot az ívet leíró három bájtnak kell követnie a következőképpen:

```
0C,X-elmozdulás,Y-elmozdulás,kihajlás
```

Az *X* és az *Y* elmozdulás értéke -127-től +127-ig terjedhet, az ív görbületét meghatározó kihajlás értékének pedig szintén -127 és +127 közé kell esnie. Amennyiben az elmozdulás által meghatározott egyenes szakasz (húr) hossza *D*, és az ív felezőpontjából erre a húrra bocsátott merőleges szakasz hossza (húrmagasság) *H*, akkor a kihajlás nagysága ((*2 * H / D*) * 127). Ennek előjele negatív, ha az aktuális pozícióból az új pozícióba az ív az óramutató járásával megegyező irányban halad.



Egy félkör kihajlása 127 (vagy -127); egyben az a legnagyobb ív, amely ezzel a kóddal egyetlen ívszakaszként ábrázolható (nagyobb íveket két csatlakozó ívszakaszból készíthet). A 0 értékű kihajlás is értelmezhető: ez egy egyenes szakaszt jelöl. Megjegyezzük, hogy a 8-as kóddal megadott egyenes szakasszal egy bájtot megtakaríthat az alakleírásban.

A csatlakozó ívszakaszok kódját (00D vagy 13) nulla vagy több ívszakaszt meghatározó értékhármas követi, melyeket egy (0,0) elmozdulás zár le (jegyezze meg, hogy ez utóbbit nem követi kihajlásérték). Az S betű például az alábbi sorozattal adható meg:

```
13, (0,5,127), (0,5,-127), (0,0)
```

A nulla kihajlású szakaszokat jól használhatja az ívláncokon belül egyenes szakaszok definiálására; ez jóval hatékonyabb módszer, mint az ív rajzolását megszakítva egy egyenes szakaszt beilleszteni, majd egy újabb ívláncot kezdeni.

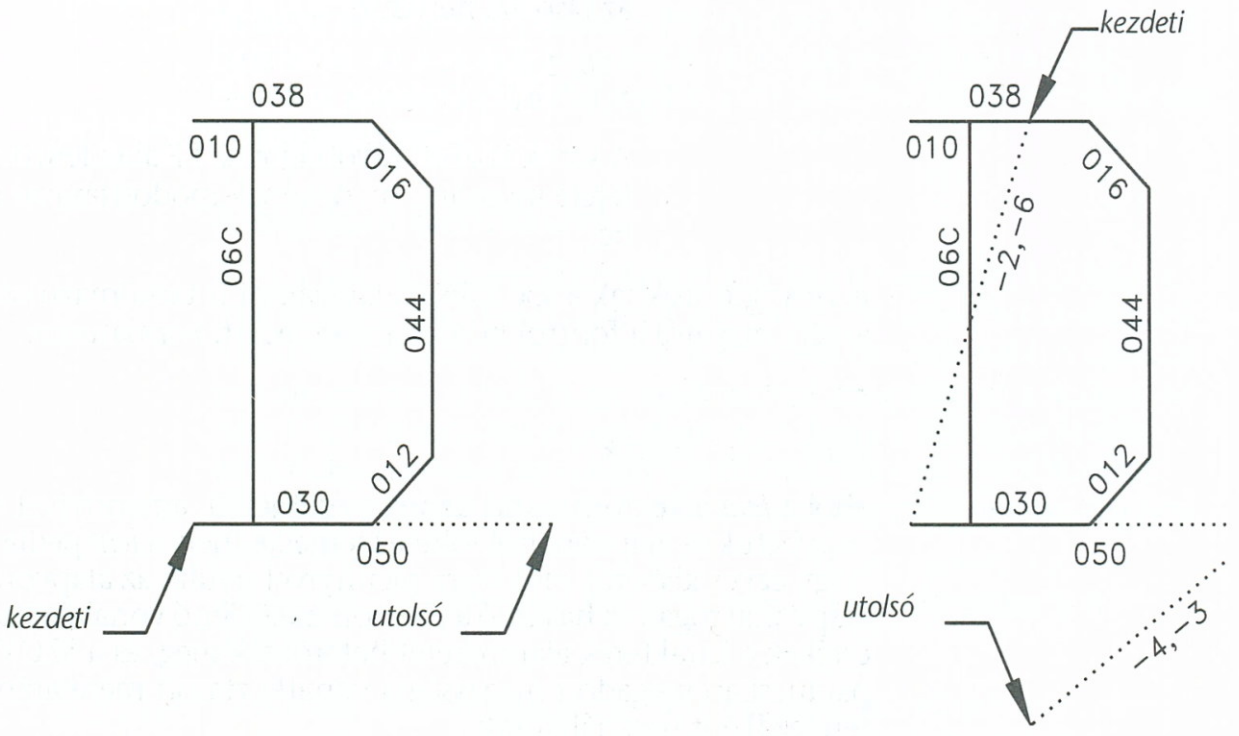
A -128 érték használata az ívszakaszok és az ívláncok definíciójában nem megengedett.

Kód 00E: Függőleges szövegre vonatkozó parancs

A 00E (14) speciális kód csak a kettős orientációjú szövegfontok leírásában használható, amikor a szövegfontot mind vízszintes, mind függőleges irányban használni kívánja. Amikor az AutoCAD egy karakterdefinícióban ezzel a speciális kóddal találkozik, a következő kódot az orientációtól függően dolgozza fel vagy ugorja át. Függőleges szöveg esetén feldolgozza, vízszintes szöveg esetén pedig átugorja a következő kódot.

A vízszintes szövegben minden karakter kezdőpontja az alapvonal bal oldali végpontja. Függőleges szöveg esetén a kezdőpont alapértelmezés szerint a karakter felső középpontját határozza meg. Normális esetben minden egyes karakter kirajzolása egy "felemelt tollal" megtett szakasszal ér véget, amely a következő karakter kezdőpontjába vezet. Ez a pont vízszintes szöveg esetén jobbra, függőleges szöveg esetén pedig lefelé található. A 00E (14) speciális kód elsődleges szerepe az, hogy a kezdő- és végpontok ezen különbségeit kezelje, és ezzel lehetővé tegye ugyanannak a karakterdefiníciónak a használatát mind a vízszintes, mind a függőleges szövegekben. Az alábbi definícióval megadott D betű például vízszintes és függőleges szövegekben egyaránt használható.

```
*68,22,nbd
2,14,8,(-2,-6),1,030,012,044,016,038,2,010,1,06C,2,050,
14,8,(-4,-3),0
```



vízszintes *függőleges*

Szövegfontok leírása

Az AutoCAD lemezeken számos szövegfont található. A STÍLUS parancs segítségével ezeket a szövegfontokat megnyújthatja, összenyomhatja, megdöntheti, azaz a karaktereket a saját kívánságának megfelelően átalakíthatja. Ezen szövegfontok segítségével tetszőleges karakternagyságú, tetszőleges szögben elforgatott alapvonalú, akár vízszintes, akár függőleges orientációjú szövegeket rajzolhat.

Amennyiben saját szövegfontokat szándékozik készíteni, javasoljuk, hogy tanulmányozza a *txt.shp* fájlt, valamint a lemezeken található többi szövegfontot, mint szemléltető példákat az itt tárgyalt témához. Az AutoCAD szövegfontjai olyan alakdefiníciós fájlok, amelyekben az alakok számozása az egyes karakterek ASCII kódjának felel meg. Az ASCII kódokat a függelék "ASCII kódok" táblázatában találja meg.

Az 1-től 31-ig terjedő kódok vezérlőkaraktereket jelölnek, amelyek közül csak egy használatos az AutoCAD szövegfontokban.

10 (LF) A soremelés (Line Feed, LF) karakter rajzolás nélkül emel egy sort. Ez ismételt SZÖVEG parancsok esetén

használatos, és arra szolgál, hogy a következő sorokat az első alá helyezze.

```
*10,5,lf
2,8,(0,-10),0
```

A sorok közötti távolságot az LF alakdefinícióban a lefelé mozgatás mértékének módosításával állíthatja be.

A szövegfontoknak a speciális 0 alakszámot is tartalmazniuk kell, amely magáról a fontról hordoz információt. Formátuma:

```
*0,4,fontnév

fent,lent,módok,0
```

ahol a *fent* a vektorhosszak azon számát jelöli, amennyivel a nagybetűk az alapvonaltól fölfelé kiemelkednek, a *lent* pedig azt jelöli, hogy egyes kisbetűk alsó szára mennyivel nyúlik az alapvonal alá. Az alapvonal fogalma hasonló a papíron megjelenő vonalakéhoz. Ezek az értékek a karakterek alapméretét határozzák meg, és a SZÖVEG parancsban megadott magasságra vonatkoztatott méretarány-tényezőként használatosak.

A *módok* bájt értéke a vízszintes orientációjú fontok esetében nulla, a kettős orientációjú (vízszintes vagy függőleges) fontok esetében pedig 2. A speciális 00E (14) parancskódot az AutoCAD csak akkor veszi figyelembe, ha a *módok* értéke 2-re van állítva.

Az AutoCAD lemezeken található fontok ezen túlmenően néhány további karaktert is tartalmaznak, amelyeket az AutoCAD méretezési szolgáltatása használ:

%%d - Fokjel (°)
%%p -Plusz/mínusz tűrés szimbóluma (±)
%%c - Kör átmérőjének méretezési szimbóluma (∅)

Ezeknek és más %%nnn vezérlőkaraktereknek a használatát az *AutoCAD on-line súgójának* "SZÖVEG" című része ismerteti.

Megjegyzés Az AutoCAD a szövegfájlfontokat az ASCII kódjuk (alakszám) alapján rajzolja meg, és nem a nevük alapján. Memóriatakarékossági okokból az alakdefinícióban az alaknevet kisbetűvel adja meg, mivel az AutoCAD a kisbetűket nem tölti be a memóriába. Például:

```
*65,11,uca
024,043,04d,02c,2,047,1,040,2,02e,0
```

Mivel az `uca` nevű alaknév kisbetűket tartalmaz, így az AutoCAD ezt nem fogja betölteni a memóriába. Ön azonban a fontdefiníciós fálj editálásakor hivatkozásként használhatja ezt a nevet; ebben a példában az `uca` a "nagybetűs A"-nak felel meg.

Speciális szövegfontok leírása

Néhány nyelv (mint például a japán) a nem-ASCII karakterek ezreit használja. Az AutoCAD egy big-fontfájlnak (nagy fontfájlnak) nevezett alakdefiníciós fájl segítségével lehetővé teszi ilyen szövegek rajzba illesztését is.

A big-font definiálása

Azok a fontok, amelyekben karakterek százai vagy ezrei találhatók, más kezelést igényelnek, mint a legfeljebb 256 karaktert tartalmazó ASCII fontok. Azon túlmenően, hogy jóval bonyolultabb technikát kell alkalmazni ahhoz, hogy egy karaktert megtaláljon a fájlban, az AutoCAD számára azt is biztosítani kell, hogy a kétbájtos kóddal ellátott karaktereket ugyanúgy kezelhesse, mint az egybájtosakat. Ezeket a problémákat a big-fontfájl elején használatos speciális kód oldja meg.

A big-font alakdefiníciós fájl első sorának a következőket kell tartalmaznia:

```
*BIGFONT n-karakterek,n-tartományok,k1,v1,k2,v2,...
```

ahol az *n-karakter* az ebben a készletben definiált karakterek számát jelöli megközelítő pontossággal; ha a tényleges szám ettől több mint 10%-kal eltér, azt a sebesség vagy a fájl mérete fogja megszenvedni. A sor hátralévő részében azokat a speciális vezérlőkódokat (escape-kódokat) adhatja meg, amelyek egy kétbájtos kód kezdetét jelölik. A japán számítógépeken például a Kandzsi-karakterek egy — a 90-AF vagy az E0-FF tartományba eső — hexadecimális kóddal kezdődnek. Ha az operációs rendszer ilyen kóddal találkozik, akkor beolvassa a következő bájtot is, és a kettőt egyetlen Kandzsi-karakterkóddá állítja össze. A *BIGFONT sorban az *n-tartomány* azt mutatja meg, hogy az escape-kódként használt számok hány összefüggő tartományba esnek. A *k1, v1, k2, v2* stb. értékek ezeknek a tartományoknak a kezdetét és a végét adják meg. Így egy japán big-fontfájl fejrészében a következőt olvashatnánk:

```
*BIGFONT 4000,2,090,0AF,0E0,0FF
```

A *BIGFONT sort követően az alakdefiníciók az AutoCAD normál szövegfontjaihoz hasonlók, azzal a különbséggel, hogy a karakterkódok (alakszámok) egészen 65535-ig terjedhetnek.

Bővített big-fontfájl definíciók

Az összetett Kandzsi-karakterek méretének csökkentése érdekében bővített big-fontfájlokat hozhat létre. A bővített big-fontok olyan alárendelt alakkódot használnak, amelyet közvetlenül egy 0 követ.

A bővített big-fontfájl első sora azonos az előzőekben ismertetett szabályos big-fontfájléval. Az ezt követő sorok formátuma viszont a következő:

```
*0,5,fontnév
karaktermagasság, 0, módok, karakterszélesség,0
   .
   .
   .
*alakszám,defbájtok,alaknév
   .
kód,0,primitív#,bázispont-x,bázispont-
y,szélesség,magasság,
   .
   .
kód,0,primitív#,bázispont-x,bázispont-
y,szélesség,magasság,
   .
   .
végjel
```

karakter-magasság A karakterszélességgel együtt jelzi a font karaktereinek definiálásához használt egységek számát.

karakter-szélesség A karaktermagassággal együtt jelzi a font karaktereinek definiálásához használt egységek számát. A karakterszélesség és a karaktermagasság határozza meg a fontot alkotó primitívek méretarányát. Ebben a környezetben primitíveknek nevezzük azokat a pontokat, vonalakat, sokszögeket, illetve karakterláncokat, amelyek a kétdimenziós térben orientált fontot alkotják. A Kandzsi-karakterek sokféle — különböző méretarányban és kombinációban ismétlődő — primitívből állnak.

módok Ugyanaz, mint a normál szövegfontok esetében.

alakszám A karakter kódja.

defbájtok Bájtméret. Minden esetben egy kétbájtos érték, amely hexadecimális vagy kombinált, decimális és hexadecimális kódokból áll.

alaknév A karakter neve.

kód Az alakleírás speciális kódja. Mindig 7 az alárendelt alak használata miatt.

primitív# Az alárendelt alak számára történő hivatkozás. Mindig kettő.

bázispont-x *X* A primitív kiindulópontjának X koordinátája.

bázispont-y *Y* A primitív kiindulópontjának Y koordinátája.

szélesség A primitív szélességének méretaránya.

magasság A primitív magasságának méretaránya.

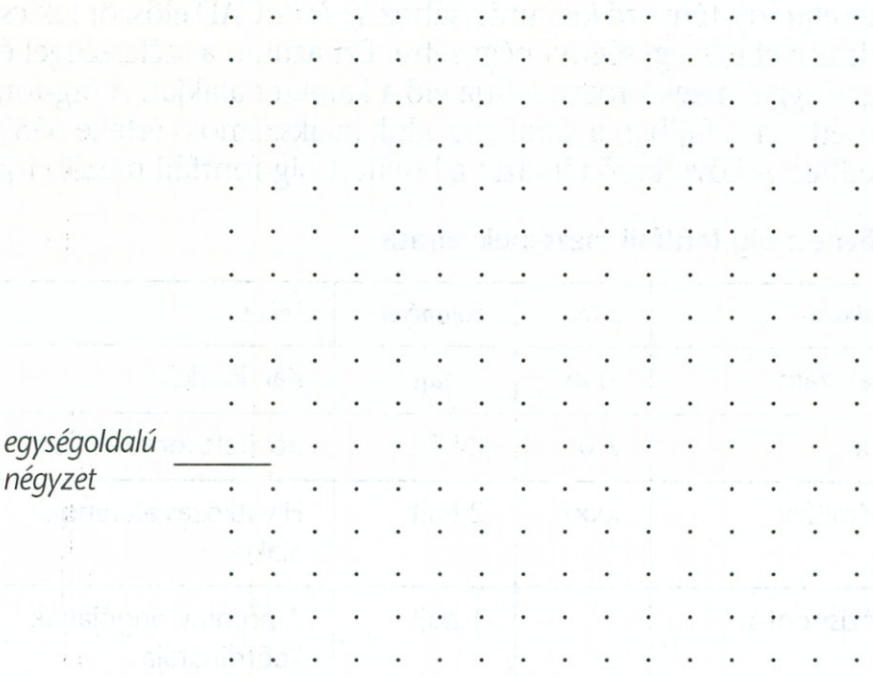
végjel Az alakdefiníció "fájl-vége" jele. Mindig nulla (0).

A méretarány-tényező kiszámításához az AutoCAD először lekicsinyíti a primitívet egy egységnyi négyzetre. Ezt azután a szélességgel és a magassággal megszorozva állítja elő a karakter alakját. A big-font alakdefiníciós fájlban a karakterkódok (alakszámok) értéke 65535-ig terjedhet. A következő táblázat a bővített big-fontfájl mezőit írja le.

A bővített big-fontfájl mezőinek leírása

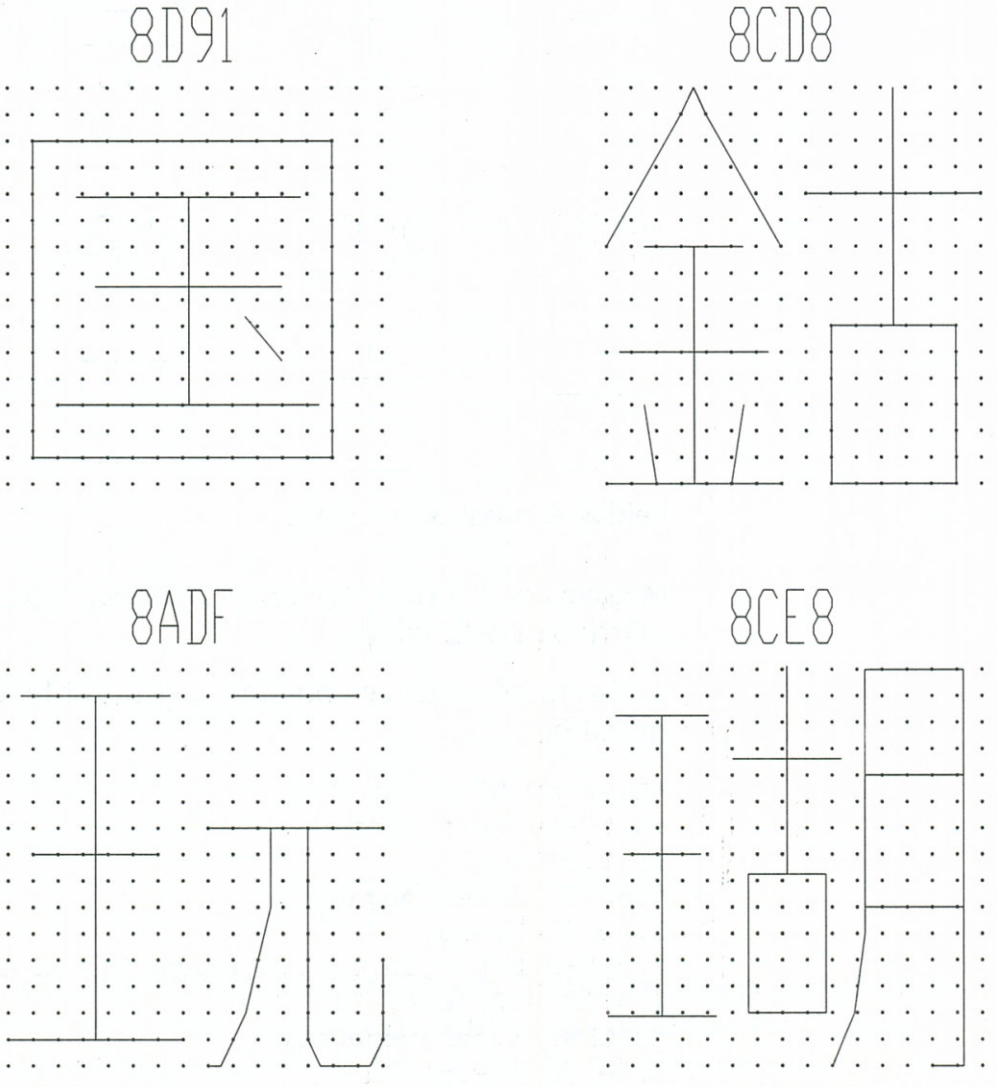
Változó	*Érték*	*Bájtméret*	*Leírás*
alakszám	xxxx	2 bájt	Karakterkód
kód	7,0	2 bájt	Bővített fontdefiníció
primitív#	xxxx	2 bájt	Hivatkozás alárendelt alakra
bázispont-x		1 bájt	A primitív origójának x koordinátája
bázispont-y		1 bájt	A primitív origójának y koordinátája
szélesség		1 bájt	A primitív szélességének léptéke
magasság		1 bájt	A primitív magaságának léptéke
végjel	0	1 bájt	Az alakdefiníció vége

A következő ábra példa egy 16 x 16-os pontmátrixra, melyet bővített big-fontok (pl. Kandzsi karakterek) tervezésére használhat. A példában a pontok közötti távolság egy egység. A kiemelt rész egy egységnyi négyzetet mutat.



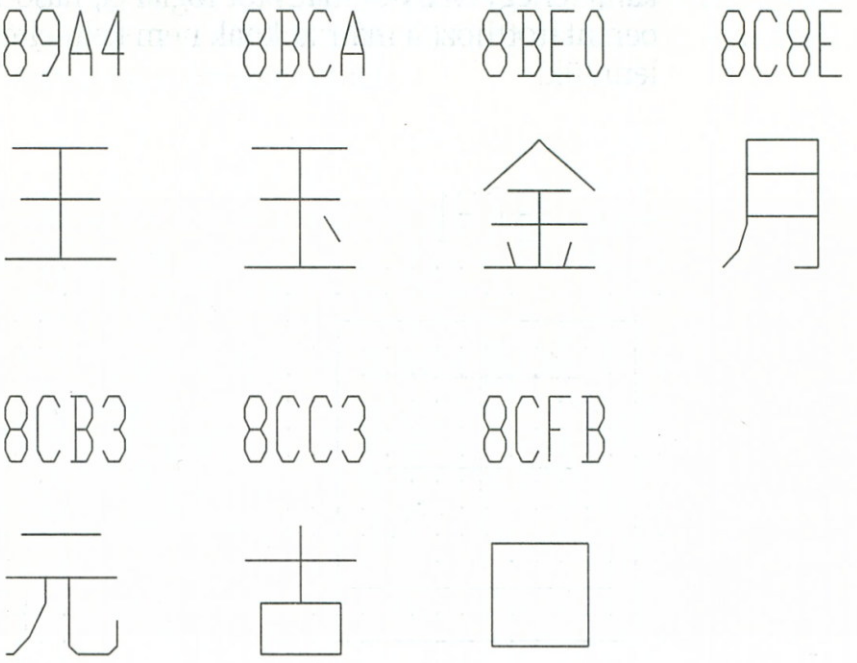
Egy Kandzsi-karakter négyzetes mátrixa

A következő ábra Kandzsi-karakterekre mutat példákat. Mindegyik karakter egy MxN-es mátrixot foglal el, hasonlóan az előző ábrán bemutatotthoz(a mátrixoknak nem kell feltétlenül négyzetesnek lenniük).



Példák Kandzsi-karakterekre

A következő ábrán Kandzsi-primitívek láthatók.



Példák Kandzsi-primitívekre

Megjegyzés A fontok egy részét nem négyzetes, hanem négyszögletes mátrixban definiálták.

A következő példa egy bővített big-fontot leíró alakdefiníciós-fájlt mutat be.

```
*BIGFONT 50,1,080,09e
*0,5,Bővített font
15,0,2,15,0
*08D91,31,nem_megadott
2,0e,8,-7,-15,
7,0,08cfb,0,0,16,16,7,0,08bca,2,3,12,9,
2,8,18,0,2,0e,8,-11,-3,0
*08CD8,31,nem_megadott
2,0e,8,-7,-15,
7,0,08be0,0,0,8,16,7,0,08cc3,8,0,8,16,
2,8,18,0,2,0e,8,-11,-3,0
*08ADF,31,nem_megadott
2,0e,8,-7,-15,
7,0,089a4,0,0,8,16,7,0,08cb3,8,0,8,16,
2,8,18,0,2,0e,8,-11,-3,0
*08CE8,39,nem_megadott
2,0e,8,-7,-15,
7,0,089a4,0,1,5,14,7,0,08cc3,5,2,5,14,7,0,08c8e,9,0,7,
16,2,8,18,0,2,0e,8,-11,-3,0
*089A4,39,primitív
2,0e,8,-7,-15,2,8,1,14,1,0c0,
2,8,-11,-6,1,0a0,2,8,-12,-7,1,
0e0,2,8,-7,13,1,0dc,2,8,11,-1,
2,0e,8,-11,-3,0
```

```
*08BCA,41,primitív
2,0e,8,-7,-15,2,8,1,14,1,0c0,
2,8,-11,-6,1,0a0,2,8,-12,-8,1,
0e0,2,0e5,1,0ec,2,063,1,8,
2,-3,2,06f,2,0e,8,-11,-3,0
*08BE0,81,primitív
2,0e,8,-7,-15,2,8,3,9,1,080,
2,8,-10,-4,1,0c0,2,8,-13,-5,1,
0e0,2,8,-7,9,1,09c,2,8,-1,14,
1,8,-6,-5,2,8,8,5,1,8,6,-5,
2,8,-11,-6,1,8,1,-3,2,8,7,3,
1,8,-1,-3,2,8,-3,15,1,01a,2,
012,1,01e,2,8,10,-14,2,0e,8,
-11,-3,0
*08C8E,44,primitív
2,0e,8,-7,-15,2,8,3,15,1,090,0fc,038,
2,8,-6,11,1,090,2,8,-9,-5,1,
090,2,096,1,0ac,8,-1,-3,01a,01a,2,8,
18,0,2,0e,8,-11,-3,0
*08CB3,61,primitív
2,0e,8,-7,-15,2,042,1,02b,02a,018,2,
0d0,1,012,034,2,069,1,01e,040,2,8,
-8,6,1,02b,2,8,4,5,1,08c,2,8,
-3,8,1,03c,2,8,-5,3,1,0e0,2,8,
-12,5,1,0a0,2,8,6,-14,2,0e,8,
-11,-3,0
*08CC3,34,primitív
2,0e,8,-7,-15,2,0c1,1,06c,0a8,064,0a0,2,8,
-5,9,1,09c,2,8,-7,5,1,0e0,2,8,
4,-11,2,0e,8,-11,-3,0
*08CFB,22,primitív
2,0e,8,-7,-15,2,0d2,1,0cc,0c8,0c4,0c0,2,8,
5,-13,2,0e,8,-11,-3,0
```

Bizonyos tervezési szakterületeken számos különleges szimbólum szöveges karakterláncokban jelenik meg. Az AutoCAD állandó szövegfontjait kibővíthetjük különleges szimbólumokkal; azonban az állandó szövegfontok kibővítésének is vannak korlátai:

- Fontfájlonként maximum 255 alak lehetséges.
- Az állandó karakterkészlet csaknem az elérhető alakok számának felét felhasználja. Csak az 1–9, 11–31, és 130–255 kódok használhatók.
- Ha többszörös szövegfontokat akar használni, akkor minden egyes fontban meg kell kettőznie a szimbólumok definícióit.
- Speciális szimbólumok használatához a %%***nnn*** kifejezést kell bebillentyűznie, ahol *nnn* a szimbólumok alakszáma. Ez igen kényelmetlen.

A big-fontok használatával ezek a problémák elkerülhetők. Escape kódként kiválaszthat egy vagy több ritkán használt karaktert (mint például (~), (|)), majd a következő karakter segítségével kiválaszthatja a megfelelő különleges szimbólumot. A következő big-font fájl használatával például a függőleges vonal (|, ASCII kódja 124) segítségével egy görög betűt, majd egy annak megfelelő latin betűt jeleníthet meg. Mivel minden karakter első bájtja 124, a karakterkódok 124x256-tal vagyis 31744-gyel tolódnak el.

```
*BIGFONT 60,1,124,124
*0,4,görög
fölött, alatt, módok, 0
*31809,n,uca
. . .    nagy alfa definíciója a "|A"segítségével
*31810,n,ucb
. . .    nagy béta definíciója a "|B"segítségével
*31841,n,lca
. . .    kis alfa definíciója a "|a"segítségével
*31842,n,lcb
. . .    kis béta definíciója a "|b"segítségével
*31868,n,vbar
. . .  függőleges vonal definíciója a " | | " segítségével
. . .
```

A big-font használata

Ahhoz, hogy a big-fontot szövegrajzoláshoz használja, létre kell hoznia egy szövegstílust a STÍLUS parancs segítségével, majd meg kell adni a big-font fájl nevét. Ugyanaz a szövegstílus képes normál ASCII fontokat is használni. Csak a két fájlnevet kell begépelnie egy vesszővel elválasztva.

Parancs: **stílus**
Szövegsílus neve (vagy ?): *név*
Fontfájl <*aktuális*>: **txt,greek...**

Az AutoCAD feltételezi, hogy az első név a normál font, míg a második a big-font.

Ha csak egy nevet ír be, akkor az AutoCAD azt normál fontnak tekinti és törli az összes vonatkozó big-fontot.

A fontfájl-nevek megadásakor kezdő- vagy végpozícióban elhelyezett vesszőkkel, megváltoztathatja az egyik fontot anélkül, hogy hatással lenne a másikra. Ezt mutatja be a következő táblázat.

A fontok megváltoztatása

Bemenet	*Eredmény*
normál, big	Mind a normál, mind a big-fontot megadja
normál,	Csak a normál fontot adja meg (a big-font változatlan)
,big	Csak a big-fontot adja meg (a normál font vátozatlan)
normál	Csak a normál fontot adja meg (ha szüséges, a big-font törlődik)
↵ (csak soremelés)	Nincs változás

Ha a STÍLUS parancs beírásával kilistázza a stílusokat vagy módosítja a már létezőket, az AutoCAD a normál font fájlt, majd vesszővel elválasztva a big-font fájlt jeleníti meg. Ha a stílus csak big-fontot tartalmaz, akkor az egy vesszőt követően jelenik meg.(például: ,görög)

Ha az AutoCAD egy karaktert keres egy szövegben, először a big-font fájlt nézi át. Ha itt nem találja a karaktert, a normál fontfájlban folytatja a keresést.

Unicode fontleírások

Az AutoCAD szabványos fontjai a hálózati főgép operációs rendszerének karaktertáblázatához igazodnak. Ez azért van így, mert a karakterek az adatbázisban ugyanolyan formában tárolódnak, mint ahogyan a billentyűzetről érkeznek. A karakterkódok egyben a fontokat is generálják. Ez akkor jelenthet problémát, amikor ékezetes, 8 bites karaktereket használunk, mert ezek kódolására sokféle szabvány létezik.

A karaktertáblák korlátai miatt szükség volt az AutoCAD által használt különböző kódlapoknak megfelelő fonthalmazokra. Ezek a fontok, bár lényegében azonosak, különböző helyen lehetnek attól függően, hogy melyik kódlaphoz definiálták őket (például a *txt8.shx* és a *txt9.shx* fájlok a DOS 850 és a ISO 8859-1 kódlapokhoz tartoznak). Ha a font kódolása nem egyezik a rajzban található szöveg fontjainak kódolásával, nem megfelelő karakterek jelenhetnek meg.

A Unicode fontok esetében, a szöveg még a kirajzolás *előtt* konvertálódik, így többé nincs szükség arra, hogy más nyelvek és környezetek számára kiegészítő fontokat hozzunk létre. Egy Unicode font - kiterjedt karakterkészletének köszönhetően - lehetővé teszi bármely nyelv vagy környezet használatát. Ez a szolgáltatás a felhasználó számára természetes, mivel a rajzok betöltésekor (a különböző kódlapoknak köszönhetően) szükség szerint az AutoCAD

kódlapjaira konvertálódnak. A rajzok mindig az AutoCAD rendszer-kódlapjára mentődnek.

Megjegyzés Az Unicode nem alkalmas minden ázsiai nyelv karaktereinek megjelenítésére. Emiatt még mindig big-fontokat használ néhány ázsiai verzió.

Fájlformátum

A Unicode alakdefiníciós fájlok mind formátumban, mind leírási módban látszólag teljesen megegyeznek a hagyományos AutoCAD alakdefiníciós fájlokkal. A fő különbség a font fejlécének leírási módjában van, amint az a következő kódban látható:

```
*UNIFONT,6,font-név
fölött,alatt,módok,kódolás,típus,0
```

A *font-név*, *fölött*, *alatt* és *módok* paraméterek ugyanazok, mint a hagyományos fontok esetében. A fennmaradó két paraméter azonban a következő módon van definiálva:

kódolás A fontok kódolása. A következő egész értékek egyike.

- **0** Unicode
- **1** Csomagolt multibájt 1
- **2** Alakfájl

típus Fontbeágyazási információ. Megadja hogy a font használati joghoz kötött-e. Az ilyen fontokat módosítani vagy kicserélni nem szabad. (bit-kódolt értékek, összeadhatóak)

- **0** A font beágyazható
- **1** A font nem beágyazható
- **2** A beágyazás csak olvasható

Az egyetlen további különbség a Unifont és a hagyományos alakdefiníció között az alakszám. Az AutoCAD által megadott Unifont alakdefiníciók, hexadecimális formátumokat használnak, szemben a decimális értékekkel. Ez nem kötelező, azonban megkönnyíti az alakszámok társítását a \U+ vezérlőkarakter-értékekkel.

A Unicode fontok használata

A program az alábbi táblázatban bemutatott formátumú vezérlőkaraktereket még a szöveg tényleges kiírása előtt Unicode alakszámokká kovertálja.

Unicode vezérlőkarakterek

Vezérlőkarakter	*Leírás*
\U+*xxxx*	Unicode alakszám. A *xxxx* az Unicode karakter hexadecimális értéke.

Unicode vezérlőkarakterek (*folytatás*)

Vezérlőkarakter	*Leírás*
\M+*nxxxx*	Multibájt alakszám. Az n számjegy az eredményként kapott multibájt kódlapazonosítóját adja meg (1: dos932, 2: big5, 3: ksc5601, 4: johab). Az *xxxx* a multibájt karakter hexadecimális értéke.

Megjegyzés Ezeket a vezérlőkaraktereket használja az AutoCAD adatbázis-kezelője is. Ilyen vezérlőkódokká konvertálódik, bármely, az aktuális kódlapon nem megjeleníthető Unicode alakszám.

Azon karakterek helyett, melyek nem találhatóak meg az aktuális fontban, kérdőjelek jelennek meg a képernyőn. Ez bármely Unicode vezérlőkarakterre is vonatkozik. Azonban a Unicode karakterek nem fordítódnak, ha az aktuális font nem támogatja a Unicode formátumot. Ebben az esetben a vezérlőkarakterek eredeti formájukban kerülnek kijelzésre.

A Unicode fontokat (ugyanúgy, mint a normál fontokat) a big-font formátummal együtt is használhatja.

A következő táblázat egy font speciális méretezési szimbólumait (a fokjelet, a plusz- és mínuszjelet és az átmérő szimbólumot) rendeli a hexadecimális Unicode-értékekhez.

Különleges szimbólumok értékei

Szimbólum	*Érték*
Fokjel	0x00B0
Plusz- vagy mínuszjel	0x00B1
Átmérő szimbóluma	0x2205

Megjegyzés A következő jeleket egyaránt használhatja Unicode, és szabályos AutoCAD fontokkal: **%%d** (fokjel), **%%p** (plusz- vagy mínuszjel), **%%c** (átmérő szimbóluma) .

A \U+ szekvenciát a normál szöveggel vegyesen is használhatja. Például:

```
"A \U+0073öveg."
```

Az előző sor a következőképpen jelenik meg:

```
"A szöveg"
```

Az `"s"` karaktert minden rendszer-kódlap tartalmazza; ezért amikor az objektum újra generálódik, a szekvencia valódi karakterré konvertálódik.

A Unicode leképezéséből hiányzó karaktereket az \M+n jelsorozat jeleníti meg. Ez elsősorban az ázsiai nyelveknél fordul elő. Ha big-font formátumot talál, a jelsorozat 16-bites értékké alakul, és ha a karakter létezik, akkor meg is jelenik a grafikus ablakban.

Azok a karakterek, melyek nincsenek leképezve a rendszer karakterkészletében, \U+*nxxxx* vagy \M+*nxxxx* utasításként, a DXF -ben jelennek meg a nem-Unicode leképezésű multibájt kódlap számára. Emellett a szimbólumtábla-nevekben elfogadottak a 8 bites és multibájt karakterek csakúgy, mint a \U+*nxxxx* és \M+*nxxxx* utasítások.

Fejlett alakdefiniáló módszer

Az AutoCAD alul- és felülírás funkciói korlátozottak. Ezek továbfejlesztéséhez azonban viszonylag egyszerű megváltoztatni az alakdefiníciós fájlokat.

Az alul- és felülírás (subscript, superscript) két lépésben történik. Először azt a "képzeletbeli tollat", kell felemelni vagy leengedni, ami a szöveget a képernyőn vektorról vektorra létrehozza. Ezután csökkenteni kell a fontméretezést. Továbbá, evvel ellentétesen kell eljárni, ha az eredeti fontot akarjuk visszakapni. Így a fontnak négy új billentyűt kell felismernie: kettőt az alulírás, kettőt a felülírás miatt. A már meglevő fontdefiníciók megváltoztatásának elkerülése végett ezeket a numerikus billentyűzeten adhatja meg.

A következő példa alapjául az AutoCAD ROMAN fontfájlja szolgált, de hasonlóan kell eljárni minden más AutoCAD eljárás esetén is. Ez az eljárás egy fonthoz négy új alakdefiníciót ad hozzá: a felső_be, a felső_ki, az alsó_be, az alsó_ki definíciókat, melyek az őket követő karakter pozícióját és méretét adják meg. Az egyszerűség kedvéért, ez a példa a szögletes- ([és]) és kapcsos zárójeleket ({ and }) újakkal helyettesíti. Ezen túlmenően más karaktereket is kicserélhet, vagy a kibővített területről (ASCII kódok 128-tól 256-ig) választhat alakszámot. Ha bővített alakszámot használ, a %%*nnn* módszert kell alkalmaznia (ahol *nnn* a karakter ASCII érétke) az új karakterek elhelyezésére.

1 Egy ASCII editorral szerkessze meg az *.shp* kiterjesztésű fájlt (ebben az esetben a *romans.shp* fájlt). Ha nem akarja módosítani az eredeti fájlt, egy új *romanss.shp* nevű fájlt is létrehozhat.

2 Keresse meg a helyettesítendő karakterek alakdefinícióját. A régi alakdefiníciókat tegye megjegyzésbe, hogy az újak elfoglalhassák helyüket. Ehhez az alakdefiníció minden sorának elejére tegyen egy pontosvesszőt. Az alakdefiníció többsoros is lehet. A szögletes zárójelek ASCII értékei 91 és 93 (05B és 05D hexadecimális értékben, ha Unicode fontokat használ). A kapcsos zárójelek ugyanezen értékei 123 és 125 (07B és 07D hex).

3 Adja össze a definíció második sorának első két értékét és az összeget ossza el kettővel úgy, ahogyan azt az alábbi példa mutatja:

```
*UNIFONT,6,Bővített Simplex Roman UNICODE-hoz
21,7,2,0
```

21 + 7 = 28, majd 28 / 2 = 14. Ezt a számot később fogjuk felhasználni.

4 A következő sorokat tegye az *.shp* fájl végére.

```
*228,8,felső_be
2,8,(0,14),003,2,1,0
*227,8,felső_ki
2,004,2,8,(0,-14),1,0
*222,8,alsó_be
2,8,(0,-14),003,2,1,0
*221,8,alsó_ki
2,004,2,8,(0,14),1,0
```

Figyelje meg a 14 és a -14 értékeket az előző sorokban. Ezek a "képzeletbeli toll" Y tengely menti eltolásának értékei. A 14 fele az ebben a fontban megadott maximális karaktermagasságnak, és jó közelítésben adja meg az alul- és felülírás mértékét.

Most mentse el a fájlt.

5 Használja a FORDÍT parancsot az *.shp* fájl befordításához.

Amint az alak és a megfelelő stílus definiálva van az új "toll fel" és "toll le" parancsokat a [,], {, és } karakterekkel adhatja ki. A [karakter aktivizálja a felülírást, a] karakter pedig visszatér a normál helyzetbe. A { karakter az alulírást helyezi üzembe, a } karakter pedig visszavonja azt.

PostScript támogatás

Az AutoCAD PostScript támogatása lehetővé teszi PostScript fontok használatát az AutoCAD rajzokban, a rajzok átadását EPS (Encapsulated PostScript) fájlként, az EPS fájlok átvételét és kétdimenziós vonalláncok PostScript mintával való kitöltését.

PostScript képek átadása

A PSKI parancs lehetővé teszi a rajz aktuális képének EPS fájlként való kimentését, amely felhasználható kiadványszerkesztéshez, illusztrációhoz vagy bemutató programokhoz.

Az alábbi részek a PSKI gazdafájl (acad.psf) felépítését és célját ismertetik. Ezen kívül további információkkal szolgálnak a fejlett PostScript kimeneti feldolgozásról.

Az AutoCAD PostScript kiegészítő fájl—*acad.psf*

Az *acad.psf* fájl (AutoCAD PostScript Support File) az AutoCAD PSKI és PSKITÖLT szolgáltatásainak fő támogatója. Ez a szerkeszthető ASCII fájl olyan részekre tagolódik, amelyek a fonthelyettesítést vezérlik, illetve meghatározzák a PostScript kitöltési mintákat, továbbá olyan *prológszekciókra*, amelyek a PostScript kódolt eljárásdefinícióit és konstansait tartalmazzák. Az *acad.psf* szekciói bármilyen sorrendben követhetik egymást, de hatékonysági okokból a `*fonts` résznek kell az első helyen szerepelnie.

Az *acad.psf* minden egyes részét egy olyan utasítás vezeti be, amely az első oszlopban egy csillagot, azután pedig a rész nevét tartalmazza. A pontosvesszővel kezdődő sorok megjegyzések, melyek az egész fájlban előfordulhatnak.

A szabvány *acad.psf* fájl az AutoCAD programhoz szállított Type 1 fontokat határozza meg, valamint az Adobe Type Manager for Windows, az Adobe Plus Pack és az Adobe Font Pack 1 összes PostScript fontját. Mivel az AutoCAD programhoz nem jár fontletöltő, módosíthatja az *acad.psf* fájlt úgy, hogy a nyomtatójában meglévő rezidens fontokra hivatkozzon. Ha nem kapott fontletöltő programot a nyomtatójához, szert tehet egyre a CompuServe Adobe fórumában. A következő részben a *acad.psf* fájlról lesz szó.

**fontok—Fonthelyettesítő tábla*

A `fonts` rész minden sora egy AutoCAD font (*.shx* vagy *.pfb*) nevével kezdődik, és egy PostScript font nevével folytatódik, amely a PSKI parancs használatakor a szövegelemek behelyettesítésére szolgál. Az Adobe által definiált PostScript fontnevekben a kis- és nagybetűk nem egyenértékűek, de az AutoCAD fontok nevében igen. A PostScript fontneveket nem ellenőrzi a program. Ha egy használni kívánt font nyomtatáskor nem elérhető, a legtöbb PostScript művelet figyelmeztető üzenetet ad ki és egy olyan alapfonttal helyettesít, mint például a Courier.

Az *acad.psf* fájlból vett alábbi minta az AutoCAD alak-/fontfájlok és a PostScript fontok egymáshoz rendelését mutatja. Az *agd.shx* megfelelője például az AvantGarde-Demi nevű PostScript font.

```
*fontok
;
agd         AvantGarde-Demi
agdo        AvantGarde-DemiOblique
agw         AvantGarde-Book
agwo        AvantGarde-BookOblique
bdps        Bodoni-Poster
bkd         Bookman-Demi
bkdi        Bookman-DemiItalic
bkl         Bookman-Light
bkli        Bookman-LightItalic
c           Cottonwood
cibt        City-Blueprint
cob         Courier-Bold
cobo        Courier-BoldOblique
com         Courier
.
.
```

Az első oszlop az AutoCAD által ismert nevet tartalmazza, míg a második a PostScript font hivatalos neve. A nevek után szereplő aláhúzások elhagyhatók; ha például az AvantGarde-Demi fontot tartalmazó fájl neve *agd_____.pfb*, akkor az első oszlopban csak `agd` szerepel.

**figureprologue—PostScript-ábra prológ*

A `*figureprologue` utasítást követő rész írja le azokat a műveleteket, amelyek a PSBE segítségével behozott PostScript ábrák EPS fájlba történő beágyazásakor (PSKI) hajtódnak végre. Ha ezt a kódot az *acad.psf* fájlban tartja, szükség szerint módosíthatja azt, hogy nem szokványos PostScript fájlokat is elfogadjon. Ez a kód kerül kiküldésre, amikor az első PSBE kép előfordul.

**isofontprologue—ISO 8859 Latin/1 szerinti újrakódolása*

Az `*isofontprologue` utasítást követő rész írja le azokat a műveleteket, amelyek a szabvány PostScript fontok újrakódolásához szükségesek, hogy kompatibilisek legyenek az ISO 8859 Latin/1 karakterkészlettel. Ez a kód akkor kerül kiküldésre, amikor először fordul elő ISO karakter egy szöveg rajzelemben.

**fillprologue—PostScript Vonallánc-kitöltési prológ*

(A `*fillprologue` utasítást követő rész adja meg azt a kódot, amely akkor kerül kiküldésre, amikor a PSKI az első PSKITÖLT paranccsal

találkozik. Vonalláncok kitöltésére, és a következő kitöltési eljárások meghatározására szolgál:

Rangefilter Általános célú eljárás, amely az előre definiált és a felhasználó által definiált kitöltési tartomány ellenőrzését végzi el.

Bármely paraméteres kitöltési minta használhatja a Rangefiltert annak ellenőrzéséhez, hogy a paraméterek az adott mintához érvényes tartományban vannak-e. A megadási mód a következő:

érték min max `Rangefilter` *eredmény*

Ha az *érték* a minimális (*min*) és a maximális (*max*) érték között van, az *eredmény* az eredeti *érték* lesz. Más esetben az *eredmény* az *érték*-hez legközelebbi határértékét veszi fel.

Vegyük például a következő Grayscale kitöltési mintát:

```
%@Fill

/Grayscale %Grayscale,1, Grayscale=50

 {

 0 100 Rangefilter 100 div 1 exch sub setgray
fill

 } bind def
```

Itt a Rangefilter azonnal lefut, és megvizsgálja a Grayscale mintának a veremben átadott paramétert. Ha az érték a tartományon kívülre esik, a Rangefilter a legközelebbi határra állítja. A mi példánkban 0-ra (*min*), vagy 100-ra (*max*).

A Rangefilter a kitöltési mintán belül is használható, a kitöltési minta határainak dinamikus számításához.

PreFill Ez az eljárás a vonalláncok megjelenítéséhez szükséges transzformáció megállapítása után fut le. A transzformációt a CMatrix változóba menti.

DoFill Ez az eljárás közvetlenül azután fut le, amint a kitöltési útvonal elkészült; ezért a Vonallánc útvonala (PostScript nyelvre fordítva) lesz az adott útvonal. A befoglaló keretet a szabvány `Bbxxyy` változóba menti

el, ezután létrehozza a szabvány transzformációt, amelyet a kitöltési eljárás használ.

EndFill Ez az eljárás a DoFill végrehajtása után fut le. Bármilyen szüséges tisztítást elvégezhet, de jelenleg nincs feladata.

DoOutline Ez az eljárás közvetlenül az EndFill után fut le. A vonallánc körvonalát generálja. A DoOutline eljárás meghívásakor, a Vonallánc körvonala az adott útvonal, továbbá a PostScript vonalvastagság a vonallánc szélességére állítódik és a méretaránya PostScript megjelenítési területének megfelelően változik.

A vonallánc körvonalának elhagyásához tegyen egy csillagot (*) a kitöltési minta neve elé.

**fill—Saját PostScript kitöltési minták definiálása*

A `*fill` utasítást követő rész határozza meg a PostScript kitöltési mintákat. Az összes kitöltési minta leírása a `*fill` utasítás és a következő, csillaggal kezdődő sor között helyezkedik el. Új PostScript kitöltési minta megadásához az *acad.psf* fájlnak ezt a részét módosítsa.

Mindegyik kitöltési mintát a következő sor vezeti be az első oszlopban:

```
%@Fill
```

Rögtön ezután, pontosan meghatározott formában a mintát definiáló Postscript művelet első sora. Példánkban az RGBColor minta definíciója szerepel:

```
%@Fill
/RGBcolor %RGBcolor,3, Red=50, Green=50, Blue=50,
 {
 /Blue exch 0 100 Rangefilter def
 /Green exch 0 100 Rangefilter def
 /Red exch 0 100 Rangefilter def

 Red 100 div 1 exch sub
 Green 100 div 1 exch sub
 Blue 100 div 1 exch sub
 setrgbcolor fill
 } bind def
```

Az eljárás első sora a következőket tartalmazza:

```
/eljárásnév %mintanév,arg_szám, [arg_n=alapérték_n]
```

ahol a változók a következők:

eljárásnév A PostScript kitöltési eljárás belső neve. A névnek egyedinek kell lennie — ezt a nevet nem használhatja a fájlban más kitöltés, illetve általában egyetlen más PostScript eljárás se.

mintanév A mintának azt a nevét határozza meg, amelyet a PSKITÖLT parancs számára kell megadni.

arg_szám A minta argumentumainak a számát határozza meg (0 és 25 közötti egész szám).

arg_n Az *n-ik* argumentum neve. Akkor jelenik meg a promptban, amikor a mintát a PSKITÖLT paranccsal használja.

alapérték_n Az n-edik argumentum alapértelmezés szerinti értékét határozza meg az első használatkor. Az argumentumok egész vagy valós számok lehetnek; az alapérték megadása kötelező.

Amikor az AutoCAD a PSKI paranccsal átadódó rajzban először találkozik egy PostScript kitöltési mintával (melynek leírása az *acad.psf* fájlban található), kimásolja azt a kimeneti, beágyazott PostScript (*.eps*) fájlba. A "/" kezdetű sortól (`/RGBColor` az előző példában) a következő "%@" sorig (vagy a rész/fájl végéig) mindent szó szerint kiír a kimeneti PostScript fájlba.

Meghívásakor a mintakitöltési funkció a Vonallánc körvonalával, mint adott PostScript útvonallal kerül végrehajtásra. A kitöltési funkció meghívásakor a PostScript transzformáció az alábbi állapotnak megfelelően lesz beállítva:

- Az árnyékolási transzformáció úgy lett beállítva, hogy a kitöltés során az alapegység a hüvelyk ezredrésze legyen.
- A **`Bbllx`**, **`Bblly`**, **`Bburx`** és **`Bbury`** változók egy ezred hüvelykben adják meg a lapon a nyomvonal befoglaló keretének kiterjedését.
- A kitöltés előtt és után egy-egy **`gsave`**, illetve **`grestore`** kerül végrehajtásra, ami biztosítja, hogy a grafikus állapotban létrejött változások nem befolyásolják a soronkövetkező elemek későbbi feldolgozását.
- A kitöltési eljárás argumentumai az *acad.psf* fájlban történt meghatározásuk sorrendjében szerepelnek a veremtárolóban — az utolsó változó a verem tetején, az utolsó előtti alatta és így tovább. A kitöltési eljárás az összes argumentumot eltávolítja a veremből.

Az első PostScript kitöltési eljáráskor az *acad.psf* fájl `*fillprologue` része kimásolódik a kimeneti *.eps* fájlba. Ez a rész olyan eljárásokat definiál, amelyeket bármely más kitöltési eljárás felhasználhat.

PSPROLOG—Saját prológrész létrehozása

A PSPROLOG rendszerváltozó arra utasítja a PSKI parancsot, hogy figyeljen az Ön által az *acad.psf* fájlban létrehozott további prológrészre. Ha PostScript kódokat helyez egy saját prológrészbe a PostScript kimenetet többféleképpen szabályozhatja. Ez hasznos lehet, ha például különböző vonalvastagságokat különböző színekhez vagy speciális vonaltípusokhoz kíván rendelni a PostScript `setdash` funkcióval. Egyetlen prológrész több PostScript funkciót tartalmazhat.

Saját prológrész létrehozásához editorral vagy szövegszerkesztővel adjon egy új részfejlécet (header) az *acad.psf* fájlhoz. A fejléc egy csillaggal kezdődő egyszerű sor, mint például:

```
*vonalvastags
```

Ebben a példában a `*vonalvastags` sor és az *acad.psf* következő része közötti minden, megjegyzést nem tartalmazó sor kikerül a PostScript fájlba.

Jóllehet tetszõleges számú prológrészt adhat az *acad.psf* fájlhoz (például mindegyik nyomtatójához egyet), ezek közül a PSKI csak egyet vesz figyelembe. Az aktuális prológ kiválasztásához használja a PSPROLOG rendszerváltozót.

A PSPROLOG értékként karakterláncot fogad el. Ez a karakterlánc a prológrész neve, hivatkozásként szerepel. A következő példa a korábban megadott `*vonalvastags` prológra hivatkozik.

a kiválasztott prológ nevét. Például:

Parancs: **psprolog**

PSPROLOG új értéke, vagy . ha semmi <*alapérték*>: **vonalvastags**

Az *acad.psf* fájl mintaképpen tartalmaz egy `*sampleprolog` nevű, magyarázatokkal ellátott prológot.

Fejlett PostScript feldolgozások

A PSKI parancs az AutoCAD rajz struktúráját az általa létrehozott PostScript fájlba kódolja. Ezt az információt igényes PostScript-alapú alkalmazások fel tudják használni az eredeti AutoCAD objektumok tulajdonságain alapuló speciális feldolgozáshoz.

Fóliaváltások

A következő hívás jelöli mindegyik fóliaváltást:

```
(fólianév) ACADLayer
```

Ez megváltoztatható úgy, hogy a felhasználó tetszőleges műveleteket hajtson végre az argumentumként megadott fólianév alapján.

Színváltások

A színek váltását a következő hívás jelöli:

```
<színszám> <vörös> <zöld> <kék> ACADColor
```

ahol a változók definiálása a következő::

színszám	Az AutoCAD által használt színszámot adja meg
vörös, zöld és kék	Megadja az alapszín összetevőit (0-1-ig) az AutoCAD 256 színű palettáján

Alapértelmezésben ez a hívás a PostScript kimeneti színt a **`setrgbcolor`** operátorral állítja be, de tetszés szerint megváltoztatható.

Vonaltípusok

Az aktuális vonaltípusban történt változásokat a következő hívás jelöli:

```
(vtípusnév) [ pont/vonal_leírás ] (igazítás) ACADLtype
```

ahol a változók:

vtípusnév	A vonaltípus neve (FOLYAMATOS, SZAGGATOTT stb)
pont/vonal_elrendezés	Megadja a "toll le" és "toll fel" vonalszakaszok hosszát az AutoCAD vonalfont definíciójának megfelelően
igazítás	Egy betű, ami megadja a mintának, az objektumhoz való igazításának módját

Ezt a hívást bármilyen PostScript vonalfont emulálásához használhatja; az argumentumként átadott AutoCAD vonaltípus-jellemzők kapcsolják. Az *acad.psf* fájl `*sampleprolog` részében talál példarutinokat az ACADLayer, ACADColor és ACADLtype funkciók használatára.

PostScript képek beolvasása

A következő rész azt a PostScript szolgáltatást ismerteti, amely képessé teszi az AutoCAD programot PostScript képek beolvasására a PSBE parancs segítségével.

Az AutoCAD PostScript értelmező—*acadps*

Az AutoCAD PostScript értelmező (*acadps*) egy ADS alkalmazás, amely a PostScript képeket AutoCAD objektumokká konvertálja. Az *acadps* program az **ads_entmake()** segítségével generálja az objektumokat, s nem egy hardver eszköz vezérlésével. Amikor egy PostScript grafikát a PSBE parancs segítségével beilleszt, az *acadps* a beállított minőségindex függvényében fordítani kezd (a minőségindex a PSQUALITY rendszerváltozóval állítható be, az *AutoCAD on-line*

súgójában a "Rendszerváltozók" című fejezetében leírtak szerint.). Ha a minőségindex értéke 0, akkor az *acadps* hívása elmarad, és a PostScript kép csak a befoglaló keretével és fájlnevével jelenik meg.

Ha a minőségindex értéke nem nulla, akkor a PSBE automatikusan betölti az *acadps* alkalmazást. A memória jobb kihasználása érdekében az *acadps* nem marad végleg az AutoCAD programban; a PSBE végrehajtása után kitörlődik a memóriából, és minden alkalommal újra betöltésre kerül. Ez elég gyorsan történik, de ha kívánja ennek elkerülésére az **xload**-dal töltse be az *acadps*-ot, adja az *acadps*-ot az *acad.ads*-hez, vagy módosítsa az *acad.lsp*-t úgy, hogy az az **xload**-dal töltse be az *acadps*-ot.

A helyes működéshez az *acadps* alkalmazásnak meg kell találnia az AutoCAD könyvtárkeresési útvonalán az értelmező inicializáló fájljait (*acadpsd.pps, acadpsf.ps, acadpsi.ps* és *acadpss.ps*), a fonttáblafájl (*fontmap.ps*), a fontdefiníciókat, valamint a fontokat .

Az *acadps* alkalmazás az Aladdin Enterprises Ghostscript Interpreter egy változata, és a Ghostscript General Public License alapján forgalmazható. A licenszszerződés másolata megtalálható az AutoCAD *support* könyvtárában lévő *copying.gs* fájlban. A szerződés szerint Ön bárkinek átadhatja az *acadps* programot. Az *acadps* forráskódját megkaphatja közvetlenül az Autodesktől. Az alkalmazáshoz díjmentesen juthat hozzá. A Ghostscript nem hitelesített PostScript értelmező; az Autodesk garancia nélkül forgalmazza.

A PostScript képek és kitöltések az AutoCAD adatbázisában mint bővített adatok szerepelnek és az alkalmazás nevével "AUTOCAD_POSTSCRIPT_FIGURE" azonosíthatók.

Az AutoCAD fonttábla-fájl—*fontmap.ps*

Az AutoCAD fonttábla-fájl (*fontmap.ps*) olyan katalógus, vagy fonttáblázat, amely az AutoCAD Postscript értelmező (*acadps*) által ismert összes fontot tartalmazza. A PSBE során ez az ASCII fájl rendeli a PostScript fontneveket a megfelelő fontleíró (*.pfb*) fájlokhoz. A *fontmap.ps* fájlnak tartalmaznia kell minden olyan fontot, amely megnevezésekor automatikusan betöltődik. A fontokat az AutoCAD keresési útvonalában bármelyik könyvtárba elhelyezhetjük.

Amikor a felhasználó a PSBE segítségével egy EPS fájlt olvas be, az *acadps* program a *fontmap.ps* fájlt használja az EPS fájlban megnevezett fontoknak megfelelő *.pfb* fájlok megtalálásához. Ha egy *.pfb* fájl neve megtalálható a *fontmap.ps* fájlban, de nincs meg a rendszerben, akkor az *acadps* program figyelmeztető üzenetet jelenít meg, majd az alapértelmezés szerinti fontra helyettesít. Következésképpen semmilyen baja nem származik abból, ha nemlétező fontot ad meg.

Az AutoCAD csomagban található eredeti *fontmap.ps* fájl az AutoCAD programmal szállított Type 1 fontokat tartalmazza, ezen kívül az Adobe által forgalmazott Adobe Type Manager for Windows, Adobe

Plus Pack és Adobe Font Pack 1 fontokat. Ha további fontokat vásárol, csak módosítania kell a fonttábla-fájlt, és megadni azokat is ugyanúgy, ahogyan a meglévők deklarálva vannak.

A *fontmap.ps* fájlban további információt talál, beleértve a fonthelyettesítés leírását is.

Hibák a beolvasott PostScript képekben

A PSBE paranccsal beolvasott hibás PostScript képek kezelése a *PostScript nyelvi referencia-kézikönyv* szerint történik. A végzetes hibák a PSBE parancs felfüggesztését okozzák. A kevésbé komoly hibák, például egy nem létező fontra való hivatkozás, a font behelyettesítését eredményezik.

Ha az AutoCAD nem tud árnyékolni egy érvényes PostScript képet, és azt továbbra is a rajzába szeretné illeszteni, állítsa a PSQUALITY rendszerváltozó értékét nullára, majd használja újra a PSBE parancsot. Ekkor az AutoCAD nem árnyékolja a képet, és nem végez hibakeresést sem. Ha azonban a PostScript kép hibás, akkor mindegyik PSKI paranccsal generált *.eps* fájl is hibát fog okozni a végső kinyomtatásnál.

Nincs rá garancia, hogy az AutoCAD Postscript értelmezõje pontosan lefordít minden olyan PostScript fájlt, amely egyébként egy PostScript Adobe változattal működik. Az AutoCAD PostScript értelmezője kompatibilis az Adobe PostScript nyelvvel, de vannak apró eltérések. Ha az AutoCAD nem tud betölteni egy érvényes PostScript képet, a PSQUALITY rendszerváltozó 0-ra állításával a kép ennek ellenére betölthető, de az csak a nevével és befoglaló keretével fog szerepelni az AutoCAD rajzban.

PostScript referenciák

Adobe Type 1 Font Format Version 1.1, Addison-Wesley Publishing Company, Inc., 1990.

Ghostscript General Public License, Aladdin Enterprises, 1989. A szerződés másolata az AutoCAD *fonts* könyvtárában levő *copying.gs* fájlban megtalálható .

PostScript Language Program Design, Adobe Systems Incorporated, Addison-Wesley Publishing Company, Inc.,1990. március

PostScript Language Reference Manual, Adobe Systems Incorporated, Addison-Wesley Publishing Company, Inc., 1991. április

PostScript Language Tutorial and Cookbook, Adobe Systems Incorporated, Addison-Wesley Publishing Company, Inc., 1989. május

4

SAJÁT MENÜK

A menüadaptálás különösen hasznos, ha egy adott alkalmazásra jellemző feladatot rendszeresen kell végrehajtania. Saját menüjéhez új kiválasztási pontot hozzáadva javíthatja munkája hatékonyságát. Így egy több lépésből álló feladat végrehajtását egy összetett művelet automatizálásaként, egyetlen menüpont kijelölésével indíthatja el.

A menüket *.mnu* kiterjesztésű ASCII fájlok definiálják. A felhasználó szabadon módosíthat egy létező menüfájlt (ilyen pl az *acad.mnu*), vagy létrehozhatja a sajátját. A menüfájl szövegének szerkesztésével meghatározhatja a menüpontok megjelenését és helyét. Ezután *menümakrókat* hozhat létre, amelyek egy menüpont kiválasztásakor meghatározott műveleteket hajtanak végre.

A menümakrók lehetnek a szükséges billentyűleütések egyszerű felvételei, vagy parancsok és AutoLISP vagy DIESEL programkódok összetett kombinációi. A még összetettebb menümakrók bizonyos döntési képességgel is rendelkeznek. A menümakró hasonlít a forgatókönyvhöz abban, hogy ez is parancsok sorozatát adja ki, de a forgatókönyvekben nincs döntési lehetőség és nem szakíthatók félbe a párbeszédes adatbevitel érdekében.

Ez a fejezet leírja, hogyan lehet az AutoCAD menüit kiegészítő saját menüket létrehozni.

Menüfájlok

A menüfájl egy olyan szövegfájl, amely a menümakrókat definiáló AutoCAD parancssorokat és makromegadási módot tartalmaz. A menüfájlok meghatározzák a menüterületek működését és megjelenését is.

Az alábbi, menüfájlok által definiált menüterületek minden platformon azonosak.

- Legördülő és kurzormenük
- Képernyőmenük
- Ábrazóna-menük
- Pozícionáló eszköz nyomógomb-menük
- Digitalizáló-tábla menük

A Windows rendszer további működési lehetőségeket nyújt a fenti menüterületeknek. A Windows az eddigieket a következő, menüfájlokban definiált lehetőségekkel egészíti ki.

- Eszköztárak
- Billentyűzet-gyorsítók
- Súgó- és eszköztippek
- Menücsoportok

A Windows további szolgáltatásairól a "Windows-specifikus menüjellemzők" című résznél, a 127. oldalon található további információ.

Menüfájl-típusok

Az AutoCAD által használt menüfájl típusa platformfüggő. Az AutoCAD DOS és UNIX verziója *.mnu* és *.mnx* kiterjesztésű menüfájlokat, a Windows változat *.mnu, .mnc, .mnr* és *.mns* kiterjesztéseket használ. Bár a DOS, a UNIX és a Windows változat is használ *.mnu* fájlokat, ezek általában nem felcserélhetőek. A DOS és UNIX által használt menümegadási mód a Windows rendszerben használtnak kisebb része; a DOS és UNIX *.mnu* fájlok jellemzően használhatóak Windows alatt, de fordítva ez nem lehetséges.

Az alábbi táblázat felsorolja a DOS és UNIX által használt menüfájlokat.

DOS menüfájlok

Menütípus	*Leírás*
.mnu	Minta menüfájl
.mnx	Lefordított menüfájl

Az alábbi táblázat a Windows által használt menüfájlokat tartalmazza.

Windows menüfájlok

Menütípus	*Leírás*
.mnu	Minta menüfájl
.mnc	Lefordított menüfájl. Ez a bináris fájl tartalmazza a menü működését és megjelenítését definiáló parancssorokat és menümegadási módot
.mnr	Menü forrásfájl. Ez a bináris fájl tartalmazza a menü által használt bittérképeket
.mns	Forrás menüfájl (Az AutoCAD generálja)

A *menü* LISP fájlok kiterjesztése *.mnl*. Ezek a DOS, UNIX és Windows által használt fájlok egy ugyanilyen nevű menüfájl betöltődésekor kerülnek be a memóriába. Ezek a fájlok a menüfájl által használt AutoLISP kifejezéseket tartalmazzák.

A *menüdefiníciós* fájlok kiterjesztése *.mnd*. Ezek makrókat tartalmazó különleges menü forrásfájlok. Az *.mnd* típusú fájlnak a *minta* könyvtárban található *mc* végrehajtható fájllal együtt kell lefordítódnia.

Menüfájlok betöltése

A menüfájlokat DOS, UNIX és Windows környezetben eltérően kezeli a rendszer. A DOS, a UNIX és a Windows a MENÜ parancsot használja egy új menü betöltésére. A Windows két másik parancsot (MENÜBETÖLT, MENÜTÖRÖL) használ kiegészítő (ún. *részleges*) menük betöltésére és törlésére, és különálló legördülő menüknek a menüsorhoz való hozzáadására ill. onnan való eltávolítására. Az alábbi alfejezetek ezeket az eljárásokat írják le.

Menük betöltése DOS és UNIX környezetben

Az AutoCAD minden egyes rajzfájlban feljegyzi az utoljára használt menü nevét, és a rajz megnyitásakor automatikusan betölti azt. Az AutoCAD egy adott fájl megtalálásához és betöltéséhez az alábbiakban vázolt lépéseket hajtja végre. Ezek a lépések hajtódnak végre akkor is, amikor az AutoCAD egy új menüt tölt be a BETÖLT parancs segítségével.

1 Az AutoCAD a megadott névvel megegyező minta menüfájlt (*.mnu*) keres a normál könyvtárkeresési eljárással. Ha megtalálja, végrehajtja a következő lépést, ha nem, kihagyja azt.

2 Ha megtalálta az *.mnu* fájlt, akkor egy ezzel megegyező nevű lefordított (*.mnx*) menüfájlt keres ugyanabban a könyvtárban. Ha a megfelelő *.mnx* fájlt megtalálta, és annak keletkezési dátuma, valamint időpontja azonos az *.mnu* fájléval, illetve annál újabb, akkor az AutoCAD ezt a fájlt fogja betölteni. Egyéb esetben az AutoCAD az *.mnu* fájlt újra lefordítja, és ebből egy *.mnx* fájlt készít ugyanabban a könyvtárban. Folytatás a 4. pontban.

3 Ha az 1. lépésben nem talált *.mnu* fájlt, akkor egy lefordított (*.mnx*) fájlt keres a megadott névvel, követve a normál könyvtárkeresési eljárást. Ha talál ilyen fájlt, akkor azt betölti. Máskülönben az 5. lépésre ugrik.

4 Miután a menüfájlt megtalálta, lefordította és betöltötte, az AutoCAD egy Menü Lisp fájlt keres. A Menü Lisp fájl neve megegyezik a menüfájl nevével, kiterjesztése pedig *.mnl* (az *.mnu* vagy *mnx* helyett). Ha megtalálta, akkor a benne található AutoLISP kifejezéseket értékeli ki ennek a fájlnak a segítségével.

5 Ha az AutoCAD sem *.mnu*, sem *.mnx* fájlt nem talál a könyvtárkeresési útvonalon, akkor ezt egy hibaüzenettel jelzi, és egy másik fájlnév megadását kéri.

Az *acad.mnl* fájl az *acad.mnu* szabvány menüfájl által használt AutoLISP programkódot tartalmazza, amelyet az AutoCAD az *acad.mnu* fájl betöltésekor minden alkalommal automatikusan betölt. Ezeknek az AutoLISP rutinoknak egy része a (`menucmd "P1.n=~"`) alakú AutoLISP függvényhívás segítségével módosítja a szabvány menü megjelenését és működését (További tudnivalók a "Hivatkozás egy legördülő vagy kurzor-almenüre" című résznél, a 117. oldalon és a **"menucmd"** címszó alatt a 13. fejezetben találhatók). Ezek a függvényhívások explicit módon változtatják meg az adott helyen található menüpontot, függetlenül annak tartalmától.

Menük betöltése Windows alatt

Az AutoCAD által a menük betöltésére használt eljárás a Windows alatt hasonló a DOS és UNIX alatt használttal, a következő kivételekkel. Az *.mnx* fájl létrehozása helyett az AutoCAD for Windows az .mnu fájlt *.mnc* és *.mnr* fájlokká fordítja. Az *.mnc* az *.mnu* fájl lefordított alakja, és az *.mnr* fájl tartalmazza a menü által használt bittérképeket. Az AutoCAD az *.mnc* fájl létrehozásával egyidőben generál egy *.mns* fájlt is. Ez egy ASCII fájl, ami kezdetben megegyezik az *.mnu* fájllal (megjegyzések vagy különleges formázás nélkül). Valahányszor a felhasználó a programozói felületen keresztül megváltoztatja a menüfájl tartalmát (pl. ilyen eset az eszköztár módosítása), az AutoCAD módosítja az *.mns* fájl tartalmát.

Bár az eszköztárak kezdeti elhelyezése meg van határozva az *.mnu* fájlban, a mutat/elrejt és a rögzített/lebegő állapotban, vagy az eszköztárak helyzetében beálló változásokat a rendszer az acad.ini fájlban rögzíti. Egy *.mns* fájl létrejötte után ez a jövőbeni *.mnc* és *.mnr* fájlok létrehozásának forrásaként szolgál. Ha az *.mnu* fájlt egy *.mns* fájl létrejötte után változtatja meg, akkor ki kell törölnie az *.mns* és *.mnc* fájlokat azért, hogy az AutoCAD új menüfájlokat generáljon és felismerje a változtatásokat. Ha az eszköztárakat a programozói felületen keresztül módosította, akkor az *.mns* fájl megváltozott részeit még annak törlése előtt ki kell vágnia és be kell másolnia az *.mnu* fájlba.

Az AutoCAD for Windows az alap- és részleges menük elvét használja. Az alapmenü a rajzzal együtt, vagy a MENÜ paranccsal betöltődő menü. A részleges menü a MENÜBETÖLT paranccsal betölthető bármely egyéb menü. A MENÜBETÖLT parancs által kizárólagosan használt menüterületek a POPn, ESZKÖZTÁR, HELPSTRINGS pontok. Ezeket a pontokat a "Windows-specifikus menüjellemzők" című résznél, a 127. oldalon tárgyaljuk. A MENÜBETÖLT parancs betölti a részleges menüt, és segítségével a menüsor legördülő menüi átrendezhetőek és kiegészíthetőek. Egy részleges menü betöltésekor az AutoCAD az *.mnc, .mnr* és *.mns* fájlok létrehozására a fentiekben már leírt eljárást használja. Ha létezik kapcsolódó .mnl fájl, akkor az AutoCAD azt is betölti.

Alkalmazási és képernyőmeghajtó menük

Sok képernyőmeghajtó program rendelkezik olyan menükkel, amelyekkel a speciális szolgáltatásai elérhetőek. Kezdő felhasználóknak nem ajánlott az eladók által támogatott menük ezen részeinek átírása, mert sok funkciójuk esetleg nem dokumentált. Ezek a menük többnyire a szabvány AutoCAD menü pillanatnyilag kiadott verzióján alapulnak és a változások általában egyetlen legördülő menüre, vagy képernyőmenük egy kis csoportjára korlátozódnak. Ezen kívül néhány eszköz a hardveren alapuló, eltérő típusú menüt is nyújt, amely esetleg

nem a szabvány AutoCAD menü- vagy menümegjelenítési lehetőségeket használja.

További információkért lapozza fel a gyártó dokumentációját.

A menüfájl felépítése

A menüfájl a különböző menüterületekhez tartozó *szakaszokra* oszlik. Ezek a szakaszok utasításokat tartalmaznak a menüben szereplő választási lehetőségek megjelenésére és funkciójára vonatkozóan.

A menü szakaszai *almenüket* foglalhatnak magukba, amelyek a rájuk történő hivatkozással szükség szerint megjeleníthetők, s így látszólag korlátlan számú menü használatát teszik lehetővé. Az egyes szakaszokon belül a parancssorozatokat és makrónyelvi elemeket (amelyek a menüből történő választáskor bekövetkező műveletet definiálják) *menümakróknak* nevezzük. Minden menümakrónak lehet címkéje. Egyes szakaszokban a menümakró címkéje megjelenik a menüterületen, míg más szakaszokban a menütípusra vonatkozó információt adja meg.

Bár szerkezetében és rendeltetésében minden menümakró hasonló, a menüpont címkéje esetében az egyes szakaszok egyedi megadási módot használnak.

Menüszakaszok

Nem szükséges, hogy a menüfájl az összes szakaszra vonatkozóan tartalmazzon bejegyzéseket. A felhasználónak csak azokra van szüksége, amelyek az aktuális alkalmazásra vonatkoznak.

A menüfájl szakaszait címkék azonosítják. Minden szakasz más-más menüterülethez tartozik, és az arra a területre szánt parancssorozatokat tartalmazza. A szakaszok címkéi és a hozzájuk tartozó menüterületek a következő táblázatban láthatók: (Az AutoCAD for Windows verzió további menüszakaszokat tartalmaz. A "Windows-specifikus menüjellemzők" című résznél, a 127. oldalon megtalálja a szakaszok listáját)

Menüszakasz-címkék

Szakaszcímke	*Menüterület*
***BUTTONS*n*	Pozícionáló eszköz nyomógomb-menüje (ahol n 1 és 4 közötti szám)
***AUX*n*	Kiegészítő eszköz menü (ahol az n 1 és 4 közötti szám)
***POP*n*	Legördülő/kurzormenü-területek (ahol az n 1 és 16 közötti szám)

Menüszakasz-címkék (*folytatás*)

Szakaszcímke	*Menüterület*
***SCREEN	Képernyőmenü-terület
***IMAGE	Ábrazóna-menüterület
***TABLET*n*	Táblamenü-terület (ahol az n 1 és 4 közötti szám)

Almenük

Egy menüszakasz meglehetősen terjedelmes lehet: sokkal több elemet tartalmazhat, mint amennyi terület a képernyőn, illetve a digitalizáló táblán rendelkezésre áll. Az *almenük* a menüpontok kisebb csoportjai egy menüszakaszon belül, amelyek a felhasználó választásának megfelelően aktivizálódnak. Minden menüszakasz különbözőképpen használja az almenüket. Ezeket a módszereket a szakaszok leírásánál, a fejezet végén vesszük sorra.

Az *almenücímke* az almenü kezdetét jelöli. Formátuma a következő:

```
**menünév kezdőszám
```

A *menünév* egy legfeljebb 33 karakterből álló karakterlánc, amely betűket, számjegyeket, valamint a speciális $ (dollár), - (kötőjel) és _ (aláhúzás) karaktereket tartalmazhatja. Az almenü címkéjének a menüfájl adott sorában önmagában kell állnia, és nem tartalmazhat beágyazott üres karaktereket. Az almenük használatának módja egy kissé eltérő minden szakaszban. Az alábbi részek a menüszakaszok közötti különbségeket írják le.

Az almenücímkéhez tartoznak a közvetlenül azt követő menüpontok, egészen a következő almenü- vagy menücímkéig, illetve a fájl végéig. Az egymást követő sorokban megadott almenünevek mindegyike a menüpontok következő csoportjára vonatkozik. Ezt almenü-nevek készítésére használhatja fel.

Egy almenü tetszőleges számú pontot tartalmazhat, de az egyes menüterületek határt szabnak az elérhető menüpontok számának. Ha például a képernyőmenü egyik almenüje 21 pontot tartalmaz, de a képernyő egyszerre csak 20 pont megjelenítésére képes, akkor az almenü utolsó pontja elérhetetlen.

Egy almenü aktivizálásához vagy kikapcsolásához az alábbi szerkezetet használhatja menüpontként:

```
$szakasz=almenü
```

`$` Az a speciális karakterkód, amely az AutoCAD rendszert egy menüszakasz betöltésére utasítja. Ennek speciális alkalmazása a `$M=` kifejezés, amely DIESEL karaktermakró hívását teszi lehetővé a menüpontból

(további tudnivaló a "Feltételes makrókifejezések" című részнél, a 101. oldalon).

szakasz A menüszakaszt határozza meg. A használható nevek:

S	A SCREEN menühöz
P0 - P16	A POP menükhöz 0-tól 16-ig
I	Az ICON menühöz
B1 - B4	A BUTTONS menükhöz 1-től 4-ig
T1 - T4	A TABLET menükhöz 1-től 4-ig
A1 - A4	Az AUX menükhöz 1-től 4-ig

almenü Meghatározza, hogy melyik almenüt kell aktivizálni. A név az éppen betöltött menüfájlban lévő egyik (** nélküli) almenücímke vagy a fentiekben megadott szakasz neve lehet.

A következő menütételek almenü-hivatkozásokat mutatnak be:

```
$S=PARTS
$T1=EDITCMDS
$T2=SCREEN
```

Az almenü aktivizálása előtt a szakasz éppen aktivizált menüpontjait az AutoCAD-be másolja egy veremtárolóba. A megnevezett almenü pontjait csak ezt követően aktivizálja. Ha például egy menüpont az alábbi parancsot adja ki:

```
$P=RÉSZEK
```

akkor az aktív képernyőtételeket bemásolja a képernyő veremtárolójába, majd a **RÉSZEK címkével jelzett almenü tételeit aktivizálja. Az előző tételek visszaállításához a menüpontnak a következő, almenü-címke nélküli kódot kell tartalmaznia:

```
$S=
```

Ezzel a legutoljára a veremtárolóba küldött pontok kiemelhetők és újra aktivizálhatók. Az egymásba ágyazott almenühívások megengedett száma 8. Amennyiben a számuk ezt meghaladja, akkor az elsőként a veremtárolóba küldött pontok elvesznek.

Az almenü-mechanizmust egy parancs használata közben, annak megszakítása nélkül is aktivizálhatja. Így az alábbi parancssorozatok egyenértékűek.

```
$S=ÍVFÉLÉK ÍV
ÍV $S=ÍVFÉLÉK
```

Mindkettő az ÍV parancsot indítja el, átkapcsol az ÍVFÉLÉK almenüre, és arra vár, hogy a felhasználó megadja az ív paramétereit. Amint a példa mutatja, az almenü-hivatkozást egy szóköznek kell követnie; ez választja el a menüpontban utána következő parancstól.

A következő minta menüfájl az almenük használatát mutatja be:

```
***KÉPERNYÕ
[Egyszerű]
```
Üres sor
```
[RAJZ... ]$S=Draw_Root
[EDIT... ]$S=Edit_Root
```
Üres sor
```
[Viszlát ]vége
```
Három üres sor, amelyek kitöltik a menüoldalt a tizedik sorig, és kioltják az almenük által megjelenített sorokat.
```
[  -FŐ-  ]$S=KÉPERNYÕ
```
Mivel az almenük egyike sem nyúlik le idáig, ez a sor minden menüben látható. Ez a menüpont újra a főmenüt aktivizálja.
```
**Draw_Root 2
```
Az almenü neve után álló 2-es szám hatására az almenü az `[Egyszerű]` *utáni sorban kezdődik.*
```
[Vonal  ]vonal
[Kör    ]kör
[Ív     ]ív
```
Legalább egy üres sor.
```
**Edit_Gyökér 2

[Radír  ]$S=Vál_obj radír
[Másol  ]$S=Vál_obj másol
[Mozgat ]$S=Obj_vál mozgat
```
Figyelje meg a menüálnevek használatát.
Legalább két üres sor, amelyek lefedik a `Vál_obj` *menü pontjait.*
```
**Obj_vál 2
**Vál_obj 2
```
Ön akár az `Obj_vál`*, akár a* `Vál_obj` *néven hivatkozhat erre a menüre.*
```
[Utolsó ]utolsó
[Előző  ]előző
[Ablak  ]ablak
[Metsz  ]metsz

[ -ELŐZŐ- ]$S=
```
A `$S=` *az előző menüt aktivizálja.*
```
***BUTTONS1
```
A pozícionáló eszköz nyomógomb-menüje.
```
;
```
Egy ↵ *lenyomását rendeli a 2. számú gombhoz.*
```
frissít
```
A FRISSÍT *parancsot rendeli a 3. számú gombhoz.*

Ebben a példában három almenü szerepel: `Rajz_Gyökér`, `Edit_Gyökér` és `Vál_obj`.

A `Rajz_Gyökér` és `Edit_Gyökér` almenük akkor aktivizálódnak, amikor Ön a fő képernyőmenüben a RAJZ, illetve az EDIT menüpontot választja ki. A `Rajz_Gyökér` almenü három választási lehetőséget

nyújt, amelyek mindegyike egy-egy AutoCAD parancsnak felel meg. Az `Edit_Gyökér` almenü ugyancsak három választási lehetőséget tartalmaz, amelyek mindegyike a `Vál_obj` almenüt hívja, mielőtt a megfelelő parancsot végrehajtaná.

A -FŐ- tétel valamennyi esetben a fő képernyőmenüt hívja elő. Egy képernyőmenü az előző képernyőmenünek csupán annyi sorát írja felül (törli), ahány sort önmaga tartalmaz. Ha egy képernyőmenü több pontot tartalmaz, mint ahány mező a képernyőn megjelenhet, illetve ha egy nyomógombmenü több pontot tartalmaz, mint ahány gomb rendelkezésre áll, akkor a felesleges pontokat az AutoCAD figyelmen kívül hagyja.

Megjegyzés A menüfájlban üres sorokkal hosszabbíthatja meg az almenüket, hogy azok lefedjék az előző menüket. Az üres sorok a fájl áttekinthetőségének javítására is felhasználhatók.

Arra vonatkozó információért, hogy a képernyő-almenük hogyan helyezhetők egymásra a `$S=X` kifejezés használatával, lapozza fel ezt a fejezetet a "Képernyőmenük" című résznél, a 106. oldalon.

Menüpontok megadási módja

Minden menüpont egy parancsból és egy paraméterből, vagy ezek sorozatából állhat. Minden menüpont általában egyetlen sort foglal el a fájlban.

Ha egy parancs paramétereit egy menüpont részeként kívánja megadni, akkor pontosan kell tudnia, hogy a parancs milyen sorrendben várja a paramétereket. Egy menüpontban minden egyes karakternek — még a szóköznek is — jelentősége van. Az AutoCAD időről időre megjelenő új, továbbfejlesztett kiadásaiban az egyes parancsok promptsorozata megváltozhat (néha maguk a parancsnevek is), ezért az AutoCAD újabb verziójára való áttérés szükségessé teheti, hogy kisebb változtatásokat Ön is elvégezzen a saját menüiben.

Egy egyszerű parancsmenü az alábbi sorokat tartalmazhatja:

```
vonal
ZOOM M
HÁLÓ
BE
HÁLÓ BE
RASZTER 0.001
```

Ha ezekből a menüpontokból egyet kiválaszt, akkor az AutoCAD ezt úgy kezeli, mintha ezeket a parancsokat közvetlenül a klaviatúráról billentyűzte volna be. Ha például a `HÁLÓ`, majd a `BE` menüpontokat választja ki, ez egyenértékű azzal, mintha a klaviatúrán a `HÁLÓ BE` karaktersorozatot billentyűzte volna be, vagy a menüből a `HÁLÓ BE` pontot választotta volna ki.

Amikor az AutoCAD a menüből kapja a parancsot, a PICKADD és PICKAUTO rendszerváltozók feltételezett értéke 1, illetve 0. Ez biztosítja a kompatibilitást az AutoCAD előző verzióival, valamint megkönnyíti a felhasználó dolgát, mivel nem kell ellenőriznie ezeknek a változóknak a beállítását.

Menüpontok címkéi

Egy menüpontnak rövid nevet vagy *címkét* adhat a képernyő-, legördülő, kurzor- vagy ábrazóna-menüben, a sor elején szögletes zárójelek közé téve a nevet, mint `[EZT]`.

A különböző típusú menük eltérő módon kezelik a címkéket. A legördülő és a kurzormenük több speciális címkézési módszert alkalmaznak. A nyomógomb-, tábla- és kiegészítő menük elején szögletes zárójelek közé tett szöveg nem jelenik meg; az csupán megjegyzések elhelyezésére szolgál a menüfájlban. További tudnivalók a "Menüpontok megjelenítésének szabályozása" című résznél, a 115. oldalon és a "Menüpont-címkék" című résznél, a 109. oldalon.

A MENUCTL rendszerváltozó

A MENUCTL rendszerváltozó a kiadott parancsoknak megfelelő képernyő-almenük automatikus lecserélését vezérli. Amikor a MENUCTL értéke 1 (Be), és valamely menüpont elindít egy AutoCAD parancsot, akkor az AutoCAD végrehajt egy `$S=parancsnév` hivatkozást (ahol a `parancsnév` a parancs neve), amely egy, a kiadott paranccsal megegyező nevű almenüt aktivizál. Az *acad.mnu* menü — a MENUCTL változót az *acad.mnl* fájlból az 1 értékre beállítva — kihasználja az ebben a szolgáltatásban rejlő előnyöket. A MENUCTL 0 (Ki) értéke befolyásolja a menü működését, de régebbi saját menük esetében előnyösebb lehet.

Menüpont lezárása

Amikor kiválaszt egy menüpontot, a parancssorozat feldolgozása előtt az AutoCAD automatikusan egy szóközt illeszt ennek a végére. Az alábbi

```
VONAL
```

menüpontot az AutoCAD úgy fogja feldolgozni, mintha a következőket billentyűzte volna be: **vonal** [Szóköz]. Bizonyos esetekben ez nem kívánatos; például a SZÖVEG vagy MÉRET parancs esetében, amikor a megadott szöveget a [↵] billentyűvel kell lezárnia, nem pedig szóközzel. Néha — éppen ellenkezőleg — egynél több szóköz (vagy [↵]) szükséges a parancs lezárásához, ám egyes szövegszerkesztők nem teszik lehetővé, hogy a sorok végéhez szóközöket illesszen. Ezeket a problémákat kétféleképpen kerülheti meg:

- A menüpontban szereplő pontosvesszőt (;) az AutoCAD a ↵ billentyűvel helyettesíti be.
- Ha egy sor vezérlőkarakterrel, azaz visszahajló törtjellel (\), plusz jellel (+) vagy pontosvesszővel (;) végződik, akkor az AutoCAD *nem* illeszt szóközt a végéhez. (A vezérlőkarakterek használatáról a menüpontok lezárásával kapcsolatban további tudnivaló a "Feltételes makrókifejezések" című résznél, a 101. oldalon található").

Példaként figyelje meg a — korábbi példában már felhasznált — `RADÍR 1` menüpontot:

```
[RADÍR 1]radír \;
```

Ha ez a pont csak a visszahajló törtjellel érne véget, akkor a RADÍR művelet nem fejeződne be, mivel az AutoCAD nem illeszt szóközt a törtjel mögé. Ezért a menüpont a pontosvesszőt használja a felhasználói adatbevitelt követően a parancs ↵ billentyűvel történő befejezésére (a visszahajló törtjel a felhasználói adatbevitelt jelöli, mint azt a következő részben ismertetjük). További példák:

```
FKR
[FKR Vlg]fkr ;
[Cím   ]szöveg \.4 0 SZERK Kft;;;1011 Bp.;;;Fő u.1.;
```

Az első tétel kiválasztásakor az `fkr` és egy Spacebar adódik át a parancssorba, és a következő prompt jelenik meg:

Origó/ZTeng/3pont/Rajzelem/Nézet/X/Y/Z/Előző/

Visszaállít/Ment/Töröl/ ?/<Világ>:

Erre a promptra a kívánt választ adhatja.

A második tétel kiválasztásakor az `fkr`, egy Spacebar és egy ; karakter (↵ billentyűként értelmezve) tűnik fel a parancssorban. Alapértelmezésként a Világ érték fogadható el. Az első és a második tétel tartalma közti különbség a képernyőn nem tűnne ki — természetesen nem is használná mindkettőt ugyanabban a menüben. A második sorban az egyszerű `FKR ;` helyett egy szögletes zárójelek közé írt címke szerepel, amellyel elkerülhető a pontosvessző megjelenése a képernyőn, és így a menüpont megjelenése is beszédesebb.

A harmadik menüpont kiválasztásakor az AutoCAD egy prompt segítségével kéri a kezdőpont megadását, majd három sorban írja ki a címet. A három pontosvessző (;;;) közül az első lezárja a szöveg karakterláncát, a második a SZÖVEG parancs megismétlését eredményezi, a harmadik pedig elfogadja az alapértelmezés szerinti elhelyezést (az előző sor alatt).

Adatbeviteli szünet

Sokszor hasznos, ha egy menüpont adatai a klaviatúráról vagy a pozícionáló eszközzel is bevihetők. Erre a célra egy visszahajló törtjelet (\) kell beillesztenie a menüpont azon pontján, ahol az adatbevitelt várja, például így:

```
[KÖR-1 ]kör \1
[FÓLIA-KI]fólia ki \;
```

A KÖR-1 szünetet tart, amíg a felhasználó megadja a kör középpontját, majd a menüből olvassa be a sugár 1 értékét. Figyelje meg, hogy a \ jelet nem követi szóköz. A FÓLIA-KI segítségével egy fólianevet adhat meg; a FÓLIA-KI azután kikapcsolja a fóliát, és kilép a FÓLIA parancsból. (A FÓLIA parancs normális esetben egy újabb prompttal további műveletek elvégzését várja, és csak úgy fejeződik be, ha a Spacebar vagy a ↵ billentyűt nyomja meg; a menüpontban a pontosvessző a ↵ billentyű megnyomásának felel meg.)

Normális esetben a menüpont az adatbevitelt követően folytatódik. Ebből következik, hogy olyan menüpont nem készíthető, amely változó számú adat bevitelét (mint például az objektumok kiválasztását) fogadja el, és ezután folytatódik. Ez alól a KIJELÖL parancs az egyetlen kivétel. Ebben, de kizárólag ebben az esetben a visszahajló törtjel addig függeszti fel a menüpontot, amíg a felhasználó teljesen meg nem határozta a kiválasztási halmazt. Példaként tekintse meg a következő menüpontot:

```
[VÖRÖSRE VÁLT]kijelöl \vált előző ;tul szín vörös ;
```

Ez a menüpont a KIJELÖL parancsot használja az egy vagy több elemből álló kiválasztási halmaz összeállításához. Ezután az Előző opcióval erre az elemsorozatra hivatkozva, a VÁLT parancs segítségével vörösre változtatja a kiválasztott rajzelemek színét.

Megjegyzés Mivel a visszahajló törtjel hatására a menüpont felhasználói adatbevitelre várva szünetet tart, ezért ezt a jelet más célra a menüpontokban nem használhatja. Ha a DOS operációs rendszerek esetén egy könyvtárhoz vezető útvonalat kíván megadni, az útvonal tagolására használja a normál törtjelet (/); például: */subdir/file*.

Az alábbi körülmények késleltetik a menüpont végrehajtását:

- Amikor az AutoCAD egy pont megadására vár, a tárgyraszter mód megelőzheti a pont tényleges megadását.
- Ha *X*/*Y*/*Z* pontszűrőt használ, a menüpont addig várakozik, amíg a teljes pont össze nem áll.
- Kizárólag a KIJELÖL parancs esetében a menüpont addig nem folytatódik, amíg a teljes kiválasztási halmaz el nem készült.

- Amennyiben a felhasználó transzparens parancsot indít el, a menüpont addig nem folytatódik, amíg a transzparens parancs be nem fejeződött, és az eredetileg várt adatbevitel meg nem történt.
- Ha a felhasználó egy másik menüpontot választ ki (egy opció kijelöléséhez vagy egy transzparens parancs használatához), akkor a felfüggesztett menüpont addig nem folytatódik, amíg a másik menüponttal nem végzett.

Idegen nyelvek támogatása a menükben

Ha olyan menüket hoz létre, amelyek az AutoCAD egy idegennyelvű változatával is használhatók, akkor az angol nyelvű szabvány AutoCAD parancsok és kulcsszavak elé az aláhúzás karaktert téve a parancsok és a kulcsszavak fordítása automatikusan megtörténik. Az *acad.mnu* fájl széleskörűen él ezzel a lehetőséggel.

A következő példa egy menü részletét mutatja:

```
[->Ív]
[3-pont]^C^C_arc
[KezdőP, KözépP, VégP]^C^C_arc;\_c
[KezdőP, KözépP, Szög]^C^C_arc;\_c;\_a
[KezdőP, KözépP, Hossz]^C^C_arc;\_c;\_l
[KezdőP, VégP, Szög]^C^C_arc;\_e;\_a
[KezdőP, VégP, Sugár]^C^C_arc;\_e;\_r
[KezdőP, VégP, Irány]^C^C_arc;\_e;\_d
[KözépP, KezdőP, VégP]^C^C_arc;_c
[KözépP, KezdőP, Szög]^C^C_arc;_c;\\_a
[<-KözépP, KezdőP, Hossz]^C^C_arc;_c;\\_l
```

Parancs megszakítása

A menüpontban a ^C^C karakterláncot használhatja annak érdekében, hogy előző, befejezetlen parancsok ne maradjanak folyamatban. Ennek eredménye azonos az Esc vagy a Ctrl+C kétszeri lenyomásával a billentyűzeten. Bár egyetlen ^C is megszakítja a legtöbb parancsot, egy MÉRET parancsból ^C^C szükséges a Parancs prompthoz történő visszatéréshez. Ezért javasoljuk a ^C^C használatát annak biztosítására, hogy az AutoCAD mindenképpen visszatérjen a Parancs prompthoz.

Az echo és a promptok

Normális esetben — a klaviatúráról bebillentyűzött adatokhoz hasonlóan — a menüpontból beolvasott karakterek is feltűnnek a képernyő parancssorában. A promptok akkor is megjelennek, ha a választ maga a menüpont adja meg. Ezt az "echo"-t letilthatja a MENUECHO rendszerváltozó segítségével (További tudnivaló a rendszerváltozók táblázatáról az AutoCAD on-line súgójában)

letilthatja. A kikapcsolt echo-t a menüpontban elhelyezett ^P segítségével tudja újra be-, illetve kikapcsolni.

Vezérlőkarakterek a menüpontokban

A menüpontnak a parancssorozatot tartalmazó részében ASCII vezérlőkaraktereket is elhelyezhet a felfelé mutató nyíl, más néven kalap (^), és egy azt követő karakter beillesztésével. A ^C karaktereket például az AutoCAD egyetlen Ctrl+C karakterré alakítja át. A nem alfabetikus vezérlőkarakterek a következőképpen adhatók meg:

```
^@      (ASCII kódja 0)
^[      (ASCII kódja 27)
^\      (ASCII kódja 28)
^]      (ASCII kódja 29)
^^      (ASCII kódja 30)
^_      (ASCII kódja 31)
```

Egy menümakróban a ^ karakter a Ctrl karakter lenyomását jelképezi.

Ezzel a módszerrel olyan menüpontokat készíthet, amelyek például ki- és bekapcsolják a hálót (a ^G segítségével), vagy megszakítják a parancsot (^C). Az alábbiakban erre láthat példákat.

```
[HálóKiBe] ^G
[*Megszakít*] ^C
```

Néha szükség van olyan menüpontokra is, amelyekkel egy vagy több karaktert adhat meg anélkül, hogy ezeket lezárt adatoknak tekintené. Az alábbiak segítségével például numerikus billentyűzetként használható menüsorozatot készíthet.

```
[1]1x^H
[2]2x^H
[3]3x^H
```

E tételek kiválasztásával a megfelelő számjegyet viszi be. Ezt valamilyen egyéb karakter (esetünkben az x betű) bevitele követi, amit a ^H szekvencia eltávolít (a Ctrl+H a "backspace" karakter ASCII kódja). E menüpontok mindegyike vezérlőkarakterrel ér véget, ezért az AutoCAD nem illeszt a végükre szóközt vagy ↵-t. Ily módon a [2], [2], [3] és [1] tételek kiválasztásával például a következő számot állítja össze: 2231. A szám beviteléhez meg kell nyomnia a ↵ billentyűt.

Az alábbi menüpontok funkciója megegyezik a fentiekével, csak ezúttal a visszahajló törtjel felhasználásával:

```
[1]1\
[2]2\
[3]3\
```

A kettő közül inkább az első módszer használatát javasoljuk. Jóllehet a második módszer megvalósítása egyszerűbb, és az esetek

többségében ugyanarra az eredményre vezet, előfordulhat, hogy egy a menü szüneteltetése közben kiadott parancs nem a várt módon fog működni.

Hosszú menüpontok

Amikor egy tétel nem fér el a menüfájl egyetlen sorában, azt a következő sorban folytathatja. Ehhez azonban a folytatni kívánt sor utolsó karaktereként egy plusz jelet (+) kell beírnia, például így:

```
[SETUP ]layer set ground-floor;;grid on; ... ;fill off;+
limits 0,0 12,9;status
```

Ez — az új rajz kezdeti értékeinek beállítására alkalmas — menüpont két sorban folytatólagosan helyezkedik el. A menüpontok tetszőleges számú soron keresztül folytathatók.

Menüpont ismétlése

Nagy a valószínűsége, hogy a kiválasztott parancsot egymás után többször is használni fogja, mielőtt továbblépne egy újabb parancsra. A legtöbben így használjuk a szerszámokat: kiválasztunk egyet, használjuk, majd választunk egy másikat, és folytatjuk a munkát. Hogy ne kelljen a "szerszámot" minden egyes használat előtt újra és újra kiválasztani, az AutoCAD lehetőséget ad az "üres válasszal" aktivizálható automatikus parancsismétlésre. Ilyenkor azonban nincs mód a parancs opcióinak megadására.

Az AutoCAD biztosítja a gyakran használt parancsok automatikus ismétlésének lehetőségét mindaddig, amíg egy másik parancsot nem választ. Azt a menüpontot, amely a `*^C^C` karakterlánccal kezdődik (közvetlenül a tétel címkéjét követően), az AutoCAD elraktározza a memóriában, és minden ezt követő Parancs promptra ezzel a menüponttal válaszol egészen addig, amíg Ön ezt — a billentyűzetről vagy egy másik menüpontból kiadott — Ctrl+C karakterrel meg nem szakítja.

Megjegyzés A "`*^C^C`" karakterlánccal kezdődő menüponton belül nem szerepelhet a `^C` (megszakít) szekvencia, mert az megszakítja a menüpont ismétlődését.

A következő részlet a parancsok kezelésének ismétléses vagy *modális* használatára mutat példát:

```
**EDITM
[Mozgat  ]*^C^CMOZGAT Single
[Másol   ]*^C^CMÁSOL Single
[Radír   ]*^C^CRADÍR Single
[Nyújt   ]*^C^CNYÚJT Single Crossing
[Forgat  ]*^C^CFORGAT Single
[Lépték  ]*^C^CLÉPTÉK Single
```

Megjegyzés A menüpont ismétlésének módszere az ábrazóna-menük tételeinél nem működik.

Az egyedi objektumkiválasztási módszer használata

Az egyedi objektumkiválasztási opció az objektumkiválasztás egyedi módját állítja be. Hatására az objektumok kiválasztására szolgáló normális párbeszéd lefolytatása helyett a következő opció által kiválasztott első objektum vagy objektumhalmaz kerül kiválasztásra. Ez a lehetőség a menüknél nagyon jól használható. Példaként tekintse meg a következő menüpontot:

```
[Radír]*^C^CRADÍR Single
```

Ez a menüpont megszakítja az aktuális parancsot, és a RADÍR parancsot az Egyedi opcióval aktivizálja. A menüpont kijelölése után rámutathat egyetlen kitörölni kívánt objektumra, vagy egy üres területen jelölve ki egy pontot ablakot nyithat. Az így kiválasztott objektumot (vagy objektumokat) az AutoCAD kitörli, a menüpontot pedig (az elején álló csillag hatására) megismétli, lehetővé téve ezzel további objektumok kitörlését. Az egyedi objektumkiválasztási módszer dinamikusabb párbeszédet eredményez az AutoCAD rendszerrel.

Feltételes makrókifejezések

A menüponton belül a $M= paranccsal vezethetők be a DIESEL nyelven írt makrókifejezések. (További tudnivaló az 5. fejezetben, az "Állapotsor konfiguráció és DIESEL karakteres kifejezési nyelv" című résznél található.) Az AutoCAD kiértékeli a tételnek a $M= után álló részét, azután az eredményt mint menüpontot vizsgálja újra. A formátuma:

```
$M=kifejezés
```

ahol a $M= az a speciális karakterlánc, amely arra utasítja az AutoCAD rendszert, hogy az utána következő karaktersorozatot DIESEL kifejezésként értékelje ki; a *kifejezés* a DIESEL kifejezés.

```
[KITÖLTKAPCS]FILLMODE $M=$(-,1,$(getvar,fillmode))
```

A KITÖLTKAPCS átvált a FILLMODE rendszerváltozó Be és Ki állapota között. Ezt úgy viszi véghez, hogy a FILLMODE értékét kivonja 1-ből, majd az eredményt visszateszi a FILLMODE rendszerváltozóba.Ez a módszer minden olyan rendszerváltozó Be és Ki állapotának átváltására felhasználható, amelynek érvényes értékei 1 és 0 lehetnek.

Ha a DIESEL karakteres kifejezésnyelvet használja "mi lenne ha" tesztek elvégézésére, felléphetnek olyan nemkívánatos körülmények, amikor a normál elválasztó szóköz vagy pontosvessző ↵ hatását

eredményezi. Ha a menüpont után ^Z karaktert ír ki, az AutoCAD az ↵ karaktert nem illeszti a makrókifejezés végéhez.

Megjegyzés A menüpontok más vezérlőkaraktereihez hasonlóan, az itt használt ^ Z karakter a ^ és a Z karakterek együttese, és nem a Ctrl+Z billentyűkombináció.

Az alábbi példákban a ^Z karaktert a rendszer a menüpontok befejezésére használja.

```
[Modeltér]^C^C_mspace $M=$(if,$(=,$(getvar,tilemode),0),$S=mview )^Z
[Papírtér]^C^C_pspace $M=$(if,$(=,$(getvar,tilemode),0),$S=mview )^Z
```

Ha ezek a menüpontok nem a ^Z karakterre végződnek, az AutoCAD szükségtelenül hozzájuk teszi a ↵ karaktert, így megismételve az utoljára beírt parancsot.

További ismertetés példák találhatók az 5. "Állapotsor konfiguráció és DIESEL karakteres kifejezésnyelv" című fejezetben.

A menük megadási módjának áttekintése

Ez a táblázat a menüfájlokban alkalmazott speciális karakterekről ad áttekintést. E karakterek használatának részletesebb ismertetésére a fejezet további részében kerül sor.

Speciális menükarakterek

Karakter	*Leírás*
***	szakaszcímkét jelöl
**	Almenü-címkét jelöl
[ ]	Közrezár egy címkét: • képernyő-, legördülő és kurzormenükhöz menüpontokhoz • ábrazóna-menükhöz, menü címekhez, diafájl-nevekhez vagy szöveghez
;	A ↵ billentyű lenyomása
^M	A ↵ billentyű lenyomása
^I	A Tab billentyű lenyomása
Spacebar	Beilleszt egy szóközt; menüpontban a parancssorozatok közötti szóköz egyenértékű a Spacebar billentyű lenyomásával
\	Szünetet iktat be felhasználói adatbevitelhez
_	Lefordítja az AutoCAD parancsokat és kulcsszavakat

Speciális menükarakterek (*folytatás*)

Karakter	*Leírás*
+	A következő sorig folytatja a menümakrót (ha a legutolsó karakter)
=*	Megjeleníti a képernyőn a legfelső szinten található ábrát, legördülő vagy kurzormenüt.
* ^C^C	Ismétlődő tétel előtagja
$	Speciális karakter, amely utasítja az AutoCAD rendszert egy menüszakasz betöltésére, vagy bevezet egy feltételes DIESEL makrókifejezést ($M=)
^B	Be-/kikapcsolja a rasztermódot (Ctrl+B)
^C	Törli a parancsot (Ctrl+C)
^D	Be- és kikapcsolja a koordináták kijelzését (Ctrl+D)
^E	Átvált a következő izometrikus síkra (Ctrl+E)
^G	Be- és kikapcsolja a Hálót (Ctrl+G)
^H	Lenyomja a "backspace" billentyűt
^O	Be-/kikapcsolja az Ortó módot (Ctrl+O)
^P	Be- és kikapcsolja a MENUECHO rendszerváltozót
^Q	Minden promptot, státuszlistát és billentyűzet-bevitelt továbbít a printernek (Ctrl+Q)
^T	Be- és kikapcsolja a tábla módot (Ctrl+T)
^V	Megváltoztatja az aktuális nézetablakot (Ctrl+V)
^Z	Nulla karakter, ami letiltja a Spacebar billentyű automatikus hozzáadását a menüpont végénél

Nyomógomb- és kiegészítő menük

A nyomógomb- (`BUTTONS1-4`) és a kiegészítő (`AUX1-4`) menük formailag azonosak. Használatuk az operációs rendszer által használt pozícionáló eszköz típusától függ.

Amennyiben a számítógépéhez nem tartozik rendszeregér (beépített pozícionáló eszköz), akkor az alkalmazott pozícionáló eszköz (egér vagy digitalizáló) a nyomógomb-menüket fogja használni. Ha viszont tartozik a géphez rendszeregér (például Windows NT, Macintosh vagy

SPARCstation esetében), akkor az a kiegészítő menüket, egyéb pozícionáló eszköz pedig a nyomógomb-menüket fogja használni.

A nyomógomb-menükre vonatkozó megállapítások a kiegészítő menükre is érvényesek. Az AUX1 menü működése megegyezik a BUTTONS1 menüével, és így tovább.

A menüfájl BUTTONS*n* szakasza az Ön pozícionáló eszközén található nyomógombokat vezérli. A szakasz minden sora egy nyomógombnak felel meg az egéren vagy a digitalizáló tábla pozícionálóján.

Az egyes nyomógomb-menük a következő billentyű/nyomógomb kombinációkkal érhetők el:

Billentyű/nyomógomb kombináció

Billentyű /nyomógomb kombináció	*Nyomógomb menü*
egyszerű gombnyomás	BUTTONS1
Shift + gombnyomás	BUTTONS2
Ctrl + gombnyomás	BUTTONS3
Ctrl + Shift + gombnyomás	BUTTONS4

Megjegyzés : A régebbi menükkel való kompatibilitás érdekében a BUTTONS1 és a BUTTONS1 egyenértékű. Ha azonban egy menü mind BUTTONS, mind BUTTONS1 szakaszt tartalmaz, akkor a BUTTONS1 élvez előnyt.

Nyomógomb-menük készítésével vagy adaptálásával a pozícionáló eszköz használatát hatékonyabbá és dinamikusabbá teheti. Ha jól megválasztott menüpontokkal egészíti ki a nyomógomb-menüt, ezzel személyes igényeihez igazíthatja pozícionáló eszközének használatát.

Egy nyomógomb-menü példájaként tekintse meg a szabvány *acad.mnu* BUTTONS1 szakaszát:

```
***BUTTONS1
;
$P0=*
^C^C
^B
^O
^G
^D
^E
^T
```

A BUTTONS1 szakaszcímkét követő első sor a pozícionáló eszköz pontkijelölő nyomógombja utáni első nyomógombra vonatkozik. Amennyiben a pontkijelölő nyomógomb az 1. számot viseli, akkor a pontosvessző (;) a ↵ billentyűt a pozícionáló eszköz 2. számú nyomógombjához rendeli hozzá. A menüszakasz címkéje utáni második sor a harmadik nyomógombra vonatkozik.

Megjegyzés A pontkijelölő nyomógombot nem lehet átdefiniálni. A pontkijelölő nyomógomb különbözhet az egyes pozícionáló eszközökön, amint a nyomógombok számozása is az eszköz gyártójától függ.

Minthogy a nyomógomb-menükben elhelyezett címkék a képernyőn nem jelennek meg, felhasználhatók megjegyzések céljára. Ez a példa a címketerületet a nyomógomb sorszámára utaló megjegyzésre használja fel:

```
***BUTTONS1
[2. gomb];
[3. gomb]$P0=*
[4. gomb]^C^C
```

Ebben a példában a 3. nyomógombhoz rendelt makró egy másik menüt jelenít meg a képernyőn. Ennek formátuma:

$Pn=*

ahol $ az a speciális karakterkód, amely az AutoCAD rendszert egy menüszakasz betöltésére utasítja. A *Pn* a POP*n* menüterületet jelöli. A =* az aktuálisan betöltött menüterület megjelenítésére utasítja az AutoCAD rendszert.

Ezért — az *acad.mnu* fájlból vett példánál maradva — amikor Ön a pozícionáló eszközének 3. nyomógombját megnyomja, a P0 menüterület (a kurzormenü) jelenik meg a képernyőn. A P0 menüterülethez jellemzően a menüfájl ***POP0 szakasza van hozzárendelve.

Ebben a szakaszban a hátralévő sorok a pozícionáló eszköz soronkövetkező nyomógombjaihoz rendelnek hozzá egy-egy parancssorozatot; a 4. nyomógombhoz például a ^C^C (Ctrl+C kétszer), az 5. nyomógombhoz a ^B szekvenciát (Raszter mód kapcsolója), és így tovább. Az Ön pozícionáló eszköze ebből csak annyi sort vesz figyelembe, amennyi a rendelkezésre álló nyomógombok száma.

A visszahajló törtjel speciális alkalmazása

Amikor egy menüpontot a több nyomógombbal rendelkező pozícionáló eszköz egyik gombjának lenyomásával választ ki, az AutoCAD nemcsak a nyomógomb számát, hanem a képernyő szálkeresztjének aktuális pozícióját is beolvassa. A menüfájl BUTTONS*n*

szakasza tételeinek gondos megszerkesztésével elérheti, hogy az AutoCAD ne vegyen tudomást a koordinátákról, vagy éppen ellenkezőleg, a nyomógombbal kiválasztott parancsban használja fel azokat.

Mint azt az előzőekben már ismertettük, a menüpontba beillesztett visszahajló törtjel (\) hatására az AutoCAD felhasználói adatbevitelre várva szünetet tart. A nyomógomb-menü a képernyő szálkeresztjének a nyomógomb megnyomása pillanatában érvényes koordinátáit automatikusan a felhasználó által megadott adatként kezeli. Ez azonban csak a menüpont első visszahajló törtjele esetében igaz. Ha a menüpont nem tartalmaz visszahajló törtjelet, akkor a pozícionáló eszköz koordinátáit az AutoCAD nem fogja felhasználni.

Példaként tekintse meg a következő menüpontokat:

```
***BUTTONS2
vonal
vonal \
```

Az első nyomógombhoz tartozó menüpont egy egyszerű VONAL parancsot ad ki, majd ezt követően a képernyőn felhasználói adatbevitelre várva megjelenik a szokásos Ponttól...: prompt. A második menüpont szintén a VONAL parancsot aktivizálja, de ebben az esetben az AutoCAD automatikusan beolvassa a pozícionáló eszköz aktuális koordinátáit is, és ezt a Ponttól...: promptra adott válasznak tekinti.

Képernyőmenük

A képernyőmenü-szakasz a képernyőmenü-területet vezérli. Alapértelmezésben ez tiltott. Ha aktívizálni akarja, olvassa el az AutoCAD Telepítési útmutatóját.

A ***SCREEN szakaszcímke az AutoCAD képernyőmenü kezdetét jelzi. Az itt látható almenü címkéjét a **S karakterlánc azonosítja. Érdemes ilyen egyszerű, rövid nevet használni, ha sok különböző menüpont hivatkozik erre az almenüre. A következő példa bemutatja, hogyan alkalmazható ez egy, az *acad.mnu* szabványhoz hasonló menüben:

A képernyőmenü-szakasz a fájlban *...és a képernyőn:*

```
***KÉPERNYÕ
**S
[AutoCAD]^C^C^P(ai_rootmenus) ^P
[* * * *]$S=OSNAPB
[ASE]^C^C^P(ai_aseinit_chk) ^P
[BLOKKOK]$S=X $S=BL
[MÉRET:]^C^C_DIM
[KIJELZÉS]$S=X $S=DS
[RAJZ]$S=X $S=DR
[EDIT]$S=X $S=ED
[LEKÉRDEZ.]$S=X $S=INQ
[FÓLIA...]^C^C_DDLMODES
[MODELL]$S=X $S=SOLIDS
[MNÉZET]$S=MVIEW
[PLOT...]^C^C_PLOT
[RENDER]$S=X $S=RENDER
[BEÁLLÍT]$S=X $S=SET
[FELÜLET]$S=X $S=3D
[FKR:]^C^C_UCS
[SEGÉDESZ]$S=X $S=UT

[ELMENT:]^C^C_QSAVE
**S
```

Almenücímkék

Az *almenücímke* az almenü kezdetét jelöli. Formátuma a következő:

```
**menünév kezdőszám
```

A *menünév* egy legfeljebb 33 karakterből álló karakterlánc, amely betűket, számjegyeket, valamint a speciális $ (dollár), - (kötőjel) és _ (aláhúzás) karaktereket tartalmazhatja. Az almenü címkéjének a menüfájl adott sorában önmagában kell állnia, és nem tartalmazhat beágyazott üres karaktereket. A *kezdőszám* egy egész szám, amely az almenü kezdősorát jelöli.

Az aktivizált almenü tételei az előző menü tételeinek helyét foglalják el az első pozíciótól (1. menümezőtől) kezdve, sorban az almenü utolsó tételéig. Így előfordulhat, hogy az almenü az előző menü tételeinek csak egy részét fedi le. A menüszakasz, illetve az almenü címkéje után egy számot beírva megadhatja, hogy az almenü hol kezdje el az előző menüpontok lefedését. Például:

```
**MINTA 3
```

A MINTA almenü aktivizálásakor az első két menümező nem változik, és az almenü csak a harmadik mezőtől kezdi el lefedni az előző menüt. Amennyiben negatív számot ad meg, akkor a menüpontok lefedése a

menü *végéhez* (a képernyő aljához) viszonyítva kezdődik a megadott értékkel.

Például, ha a főmenüből kiválasztja a ZOOM nevü tételt, akkor a ZOOM parancsot tartalmazó almenü fog megjelenni. A parancs almenü meghívásának egy másik módja megtalálható a "A MENUCTL rendszerváltozó" című résznél, a 95. oldalon.

Az *acad.mnu* fájlban található menük legtöbbje a 3. mezőn kezdődik, ezáltal az `[AutoCAD]` és a `[* * * *]` címkék a képernyőn maradnak.

Az fenti példában a következő sor a `**BL` almenüre hivatkozik.

```
[BLOKKOK]$S=X $S=BL
```

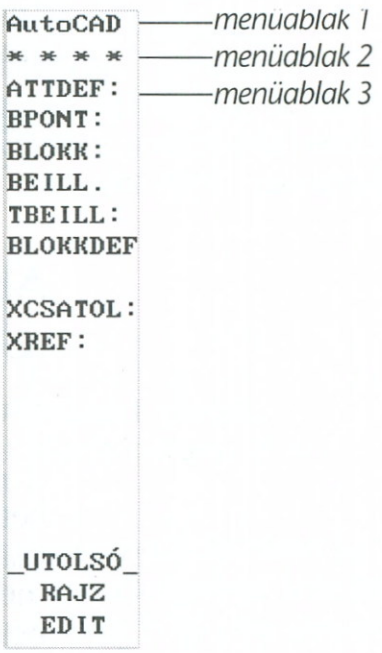
A következő példa bemutatja, hogyan jelenik meg a `**BL` almenü a képernyőn. Figyelje meg, hogy az `ATTDEF:` címkéjű első sor a képernyő 3. mezőjén jelenik meg.

A képernyőmenü-szakasz a fájlban

```
 **BL 3
[ATTDEF:   ]^C^C_ATTDEF
[BPONT:    ]^C^C_BPONT
[BLOKK:    ]^C^C_BLOKK
[BEILL:    ]^C^C_BEILL
[TBEILL:   ]^C^C_TBEILL
[BLOKKDEF:]^C^C_BLOKKDEF

[XCSATOL: ]^C^C_XCSATOL
[XREF:     ]^C^C_XREF
```

...és a képernyőn:



Az UTOLSÓ, RAJZ és EDIT menüpontok megjelennek a képernyőmenü-terület alján, mivel részei a `**X` almenünek, amelyik `$S=X` almenüre hivatkozik.

Az előző példában a következő sor először a `**X` almenüre hivatkozik a `$S=X` paranccsal.

```
[BLOKKOK]$S=X $S=BL
```

Ez az almenü letakarja az előző menü nem kívánt szövegét. A jelen példában az AutoCAD megjeleníti a `**X` almenüt, majd átlapolja azt a `**BL` almenüvel. Az eredmény a képernyőmenü-területen — a 3.

mezőtől kezdődően — az iménti példában ábrázolt módon jelenik meg.

A **X almenü, mint alább látható, a 3. mezőn kezdődik, 15 üres sort tartalmaz, és az UTOLSÓ, RAJZ és EDIT tételek címkéjével ér véget. Ez képezi az alapot, amelyet a többi menü átlapol. A **X almenü a következő:

```
**X 3
...                    15 üres sor

[__UTOLSÓ__]$S= $S=
[ RAJZ    ]^C^C$S=X $S=DR
[ EDIT    ]^C^C$S=X $S=ED
```

Az utóbbi tételek közül az UTOLSÓ visszaállítja az előző képernyőmenü tételeit (további tudnivalók az "Almenük" című részнél, a 91. oldalon). A RAJZ és az EDIT tétel egyaránt előbb a **X, majd a **DR, illetve a **ED almenüre hivatkozik.

Menüpont-címkék

A képernyőmenüben a menüpontnak tipikusan csak az első nyolc karaktere jelenhet meg, ezért a következő példában a

```
RASZTER 0.001
```

parancsból csak a RASZTER látszana. A menüpontnak rövid címet adhat, a sor elején szögletes zárójelek közé foglalva, mint [EZT]. A cím első nyolc karaktere fog a képernyőmenü megfelelő mezőjében megjelenni, a további karakterek megjegyzésként szolgálhatnak. Valahányszor ezt a menüpontot választja ki, a lezáró szögletes zárójelet közvetlenül követő karakterláncot az AutoCAD parancsként fogja értelmezni.

Példaként tekintse meg a következő menüt:

```
[FINOM ]raszter 0.001
[DURVA ]raszter 0.1
[KÉSZ  ]vége
[ABBAHAGY]kilép
```

Ebben az esetben a KÉSZ és az ABBAHAGY elnevezés egyszerű megfelelője a VÉGE, illetve a KILÉP parancsnak.

Megjegyzés A menüpontok maximális száma a képernyő méretétől függ. A képernyőmenü mezőinek számát a SCREENBOXES rendszerváltozóval kérdezheti le.

Legördülő és kurzormenük

A legördülő és kurzormenük lépcsőzetesen lebomló *láncolatot* alkotó menük (*sétáló* vagy *hierarchikus menü* néven is ismertek). Ilyenformán a menüterületek cserélgetése nélkül is lehetővé teszik a menük logikus elrendezését. A kurzormenü a gyakran használt menüpontok — mint például a tárgyraszter-módok — gyors elérését biztosíthatja. A legördülő és a kurzormenü tételei közel azonosan viselkednek más menüterületek tételeivel, továbbá a menümakrók definiálása is hasonlóan történik, mint a szabvány képernyő- vagy táblamenük esetében.

A legördülő menük legfeljebb 999 menüpontot tartalmazhatnak. A kurzormenü tételeinek száma 499-ig terjedhet. Mindkét határ az egy hierarchiában (láncolatban) szereplő valamennyi menüre együttesen értendő. Amennyiben a menüfájlban lévő menüpontok száma meghaladja ezen határokat, akkor az AutoCAD a fölös tételeket nem veszi figyelembe. Ha a legördülő vagy a kurzormenü hosszabb, mint a grafikus képernyőn függőlegesen rendelkezésre álló hely, akkor csak annyi jelenik meg belőle, amennyi a képernyőn elfér.

A legördülő menüktől eltérően, amelyek a menüsorból *gördülnek le*, a kurzormenü mindig a grafikus képernyőn látható szálkereszt helyén, vagy ahhoz közel jelenik meg. E kétféle `POP`*n* menüszakasz szintaktikájában megegyezik, azzal a különbséggel, hogy a kurzormenü címe nem szerepel a menüsorban. A legtöbb rendszer esetében a kurzormenü címe egyáltalán nem jelenik meg (egy látszólagos címet azonban mégis meg kell adni). A kurzormenü csak a `$0=*` menüparancs útján érhető el, amelyet egy másik menüpont vagy egy AutoLISP, illetve ADS program adhat ki. A kurzormenü működése közben a menüsor nem érhető el.

Megjegyzés Ha nincsenek legördülő menük definiálva (POP1 – POP16), a kurzormenü (POP0) nem működik.

Legördülő és kurzormenük megadási módja

A menüsorból legördülő menüket és a kurzormenüt a `***POP`*n* menüszakaszok vezérlik; a `POP0` a kurzormenüt, a `POP1`-től `POP16`-ig terjedőek pedig a legördülő menüket. Némelyik környezetben letilthatóak a legördülő menük, ha az Ön legördülő menüje tiltott állapotban van (az aktiválással kapcsolatban további tudnivalók az A függelék "Video képernyőopció" című részében az *AutoCAD Üzembehelyezési útmutatójában*).

Amennyiben Ön a státuszsort konfiguráláskor bekapcsolta, az AutoCAD a betöltéskor minden alkalommal végigpásztázza a menüfájlt `POP`*n* szakaszok után kutatva. A `POP1`-től `POP16`-ig terjedő szakaszokra vonatkozóan elkészíti a menüsort, amely ezen szakaszok címéből tevődik össze. A menüsor, a legördülő és a kurzormenü

szolgáltatás automatikusan elmarad, ha a menüfájlban egyetlen ***POP*n szakasz sem szerepel, vagy Ön a állapotsort konfiguráláskor kikapcsolta. Ez utóbbi esetben az AutoCAD figyelmen kívül hagyja a menüfájlban található POP*n* szakaszokat.

Az alábbi táblázat azokat a karaktereket mutatja be, amelyeknek a szögletes zárójelen belül speciális funkciójuk van a legördülő és kurzormenük címkéiben.

Különleges címkekarakterek

Karakter	*Leírás*
--	A legördülő és kurzormenükben tagolóvonallá kiterjedő tételcímke (egyéb karakterek nélkül használva).
+	A következő sorban folytatja a makrót (ha a sor utolsó karaktere).
->	Ez a címke-előtag a legördülő és kurzormenükben az almenüvel rendelkező tételt jelöli.
<-	Ez az címke-előtag a legördülő és kurzormenük almenüiben az utolsó tételt jelöli.
<-<-...	Ez a címke-előtag a legördülő és kurzormenük almenüiben az utolsó tételt jelöli, és lezárja a szülőmenüt (annyi <- szüséges, ahány szülõmenüt le kíván zárni))
$(	Lehetővé teszi a legördülő és kurzormenük cimkéjében egy DIESEL karakteres makró kiértékelését (ha a "$(" az első két karakter). (*Nem minden környezet támogatja*)
~	Ez a címke-előtag letilt (elhalványít) egy menüpontot.
!c	Ez a címke-előtag egy speciális (nem alfanumerikus) c karakterrel jelöl meg egy menüpontot a legördülő és kurzormenükben. Jegyezze meg, hogy a !. (felkiáltójel és pont) ellenőrző jelet (pipát) eredményez. Egyes rendszerek a c jelet nem veszik figyelembe és mindenképpen a pipa jelet használják, míg más rendszerek, amelyek a pipát nem támogatják, egy másik karaktert használnak helyette.
/c	A menügyorsító billentyűt jelöli egy legördülő vagy kurzormenü cimkéjében (Nem minden környezet támogatja).

Különleges címkekarakterek (*folytatás*)

Karakter	*Leírás*
<c	A karakterfont megjelenítésének stílusát vezérli egy legördülő vagy kurzormenü címkéjében. Az érvényes stílusok: <B = kövér (bold) <O =körvonalas (outline) <S = árnyékolt (shadow) <I = dőlt (italic) <U =aláhúzott (underline) (Nem minden platform támogatja.)
^*name*^	A *név* nevü ábrát jeleníti meg egy legördülő vagy kurzormenü címkéjében (Nem minden környezet támogatja)

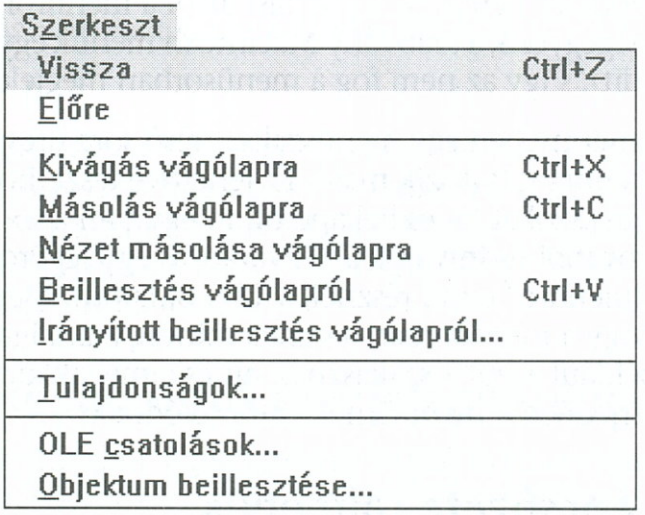
A DIESEL használata a menük címkéiben függ az adott platformtól; egyes rendszerek esetében a menüfájl kezdeti betöltése után már nem kerül sor a menücímkék aktualizálására. Az egyéb címkejelölési lehetőségek szintén platformfüggőek. A legördülő és kurzormenük címkéiben szereplő minden olyan jelet, amely nem támogatott, az AutoCAD figyelmen kívül hagy.

A következő példa egy legördülő vagy kurzormenü makrójának elkészítésénél használt megadási módot mutatja be:

A legördülő menü szakasza a fájlban

```
***POP2
[/zSzerkeszt]
[/VVissza\tCtrl+Z]_u
[/EElőre\tCtrl+A]^C^C_redo
[--]
[/KKivágás vágólapra\tCtrl+X]^C^C_cutclip
[/MMásolás vágólapra\tCtrl+C]^C^C_copyclip
[/NNézet másolása vágólapra]^C^C_copylink
[/BBeillesztés vágólapról\tCtrl+V]^C^C_pasteclip
[/IIrányított beillesztés vágólapról...]_pastespec
[--]
[/TTulajdonságok...
[--]
[/cOLE csatolások...]^C^C_olelinks
[/OObjektum beillesztése...]^C^C_insertobj
```

...és ahogyan a menüsorból legördülve megjelenik:



A legördülő menük rendszerint közvetlenül a menüsorban hozzájuk tartozó cím alatt jelennek meg. A jobb szélen elhelyezkedő menük azonban balra tolódnak, hogy a hosszú menüpontok is elférjenek. Az egy legördülő menüben elhelyezhető tételek maximális számát (minimum 21) a megjelenítő eszköz lehetőségei határozzák meg; ezt vegye tekintetbe a saját menük készítésénél.

Amennyiben a kurzormenü hívására a képernyő jobb szélének a közelében kerül sor, akkor a menü balra tolódva jelenik meg, hogy a hosszú menüpontok is elférjenek.

Címek a menüsorban

A legördülő menük esetében az első címke a menüsorban megjelenő címet határozza meg, az azt követő címkék pedig a menü és az almenük tételeit. Az előző példában a `POP2` menüterület definíciója látható. A `***POP2` szakasz címkéje utáni első sorban álló `[Eszköz]` hatására az `Eszköz` mint cím jelenik meg a menüsorban. A további, szögletes zárójelben lévő címkék, (mint például a `[súgó!]`), az Eszköz legördülő menüben és annak almenüiben jelennek meg tételként; hosszuk bármekkora lehet.

A menüsorban minden cím legfeljebb 14 karakter hosszúságú lehet; mivel azonban sok képernyő csak 80 karaktert enged meg a menüsorban, ezért a menücímek átlagos hosszát 5 karakter alatt kell tartania, ha valamennyi (16) legördülő menüterületet használni kívánja. A cím célszerűen ne tartalmazzon beágyazott üres karaktereket, mivel ez a felhasználót megzavarhatja. Amennyiben a címek nem férnek el mind a menüsorban, akkor a leghosszabbal kezdve az AutoCAD egy-egy karakterrel addig csonkolja a címeket (minden alkalommal megkeresve a leghosszabbat), amíg az összes el nem fér. Ez furcsa címeket eredményezhet, ezért, ha lehetséges, próbálja valamennyit 80 oszlopon belül elhelyezni.

Megjegyzés Az AutoCAD Macintosh és Windows változatában a rendszer nem vesz tudomást arról a menüről, amelynek címe megegyezik a szabvány Macintosh menük egyikével (például File vagy Edit), s így az nem fog a menüsorban megjelenni.

Amennyiben egy menüszakasz első sora üres, akkor ehhez nem menücím tartozik majd. Ez lehetővé teszi, hogy egész menüszakaszokat ki- és bekapcsoljon egy `$Pn=xxx` menücsere paranccsal (további tudnivalók a "Hivatkozás egy legördülő vagy kurzor-almenüre" című részнél, a 117. oldalon). Megjegyezzük azonban, hogy a menüsorban a címek mindig balra igazítva jelennek meg. Ha például a `POP3` szakasznak nincs címe, akkor a `POP4`-től `POP16`-ig terjedően a menücímek balra tolódnak.

Lépcsőzetes almenük

A legördülő és kurzormenük speciális jeleket (mint például ->, <- és <-<-...) használnak a menüláncolat hierarchiájának vezérlésére. Ezek a speciális karakterek jelzik az almenüket, az almenük utolsó tételét, és be is zárhatják az összes szülőmenüt. Valamennyi speciális karakterláncnak a menüpont címkéjének legelején kell állnia:

A -> speciális karakterek jelzik, hogy ebből a tételből almenü nyílik. Például:

```
[->Pontszűrők]
```

Ha belép az Eszköz menübe és kiválasztja a Pontszűrők menüpontot, vagy a nyíl alakú kurzort a tétel jobb oldali végének közelébe viszi, akkor a Pontszűrők almenü jelenik meg.

A <- speciális karakterek jelzik, hogy ez egy almenü utolsó tétele. Például:

```
[<-.YZ].YZ
```

A <-<-... speciális karakterek jelzik, hogy ez egy almenü, és egyben a szülőmenük utolsó tétele. Például:

```
[->Szöveg]
[->Attribútumok]
[<-<-Kigyűjt...]^C^C_ddattext
```

Menüpontok tagolása

Tagolóvonalak létrehozására két kötőjelből álló címkét használhat.

[--]

Minthogy a legördülő és kurzormenük szélességét a bennük szereplő leghosszabb menüpont határozza meg, a fenti példában látható címke a menü teljes szélességét kitöltő tagolóvonalat fog alkotni. A tagolóvonalak nem választhatók ki a menüben, és a hozzájuk rendelt makró is hatástalan.

Menüpontok megjelenítésének szabályozása

Egy további lehetőség a legördülő és kurzormenük egyéni kialakítására a tételcímkék megjelenítésének szabályozása. Letilthatja (elhalványíthatja) a címkéket, megakadályozva ezzel a kiválasztásukat, valamint megjelölheti a menüpontokat egy pipával vagy más jellel.

A címkékben DIESEL kifejezések is alkalmazhatók a címke tartalmának módosítására. Ily módon Ön a program futása közben letilthatja, illetve megjelölheti a címkéket, vagy megváltoztathatja a képernyőn megjelenő szövegüket. A "DIESEL kifejezések a menükben" című résznél, a 144. oldalon talál erre vonatkozó példákat.

Megjegyzés Bizonyos módszerek alkalmazása esetén előfordulhat, hogy a menüpontok címkéjének letiltása vagy megjelölése nem lesz összhangban az AutoCAD pillanatnyi állapotával. A menüpontok ugyanis "nem tudják", hogy Ön a menü megkerülésével nem változtatta-e meg valamelyik változót, vagy nem töltötte-e be újra a menüt a változó módosítása óta. Legjobb, ha ezekkel a lehetőségekkel akkor él, amikor egy alkalmazás csak a legördülő/kurzormenük révén érhető el, vagy megfelelő módszert biztosít a menüpontok címkéjének egy ismert alaphelyzetbe történő beállítására.

Címkék letiltása

A hullám (~) jellel kezdődő címkék elhalványítva jelennek meg. Ez egyezményesen azt jelenti, hogy a tétel jelenleg nem választható ki. A tételhez esetlegesen kapcsolódó bármilyen parancs vagy almenü hatástalan, illetve nem érhető el.

Ezek a menücímkék eredendően halványan jelennek meg:

```
[~Vonal]
[~->Vlánc]
```

A halványan megjelenő tételből elágazó almenü nem érhető el.

A menüpontok címkéi tartalmazhatnak DIESEL kifejezéseket, amelyekkel a címke minden egyes megjelenítésekor eldönthető, hogy a menüpont kiválasztható legyen-e vagy sem. További tudnivalókat a

"DIESEL kifejezések a menükben" című résznél, a 144. oldalon találhat.

Az ezen a tételcímkén belül lévő DIESEL kifejezés letiltja a címkét arra az időre, amíg egy parancs aktív:

```
[$(if,$(getvar,cmdactive),~)MOZGAT]mozgat
```

Címkék megjelölése

Egy menüpontot megjelölhet egy előtte álló ellenőrző jellel (pipával), ha a címke elejére egy felkiáltójelet és egy pontot (!.) illeszt. Ugyanezt elérheti speciális, (számtól és betűtől különböző) jellel, ha a címkébe a felkiáltójel után azt a karaktert írja, amelyet a menüpont előtt szeretne látni (például !*c*, ahol a *c* a kívánt karakter). Így lehetséges egy szerkesztési mód Be-/Kikapcsolt állapotának jelzése. Egy menüpont megjelölése nincs hatással annak kiválaszthatóságára, a jellel ellátott tétel azonban el is halványítható.

Lehetnek olyan rendszerek, amelyek nem támogatják a pipa jelet (✓). Ebben az esetben az AutoCAD egy másik karaktert fog helyette használni.

A formátum a következő:

```
[!c címkeszöveg]
```

ahol

! a speciális karakterkód, amely utasítja az AutoCAD rendszert a menüpont megjelölésére; c pedig a jelölésre használt speciális (nem alfanumerikus) karaktert jelöli. Egyes rendszerek a *c* karaktert nem veszik figyelembe, és mindenképpen a pipa jelet feltételezik. A *címkeszöveg* a menüpont címkéjének a szövege. Például:

```
[!.Vonal]
```

A pont egy pipát jelent, így a Vonal menüpont előtt egy ✓ fog állni.

A menüpontok címkéi tartalmazhatnak DIESEL kifejezéseket, amelyekkel a címke minden egyes megjelenítésekor eldönthető, hogy a menüpont meg legyen-e jelölve vagy sem. Lásd a "DIESEL kifejezések a menükben" című részt a 121. oldalon.

A következő példa egy pipát helyez azoknak a tételcímkéknek a bal szélére, amelyek éppen be vannak kapcsolva:

```
[$(if,$(getvar,orthomode),!.)Orto]^O
[$(if,$(getvar,snapmode),!.)Raszter]^B
[$(if,$(getvar,gridmode),!.)Háló]^G
```

Egyidejű letiltás és megjelölés

A menüpontok egyidejűleg megjelölhetők és el is halványíthatók. Ennek formátuma:

```
[~!c címkeszöveg]
```

vagy

```
[!c~ címkeszöveg]
```

ahol:

A ~ az a speciális karakterkód, amely utasítja az AutoCAD rendszert a menüpont elhalványítására; ! az a speciális karakterkód, amely utasítja az AutoCAD rendszert a menüpont megjelölésére; a c pedig a jelölésre használatos (nem alfanumerikus) karaktert jelöli. Egyes rendszerek a *c* karaktert nem veszik figyelembe, és mindenképpen a pipa jelet feltételezik. A *címkeszöveg* a menüpont címkéjének a szövege. Például:

```
[~!.Vonal]
```

A pont egy pipát jelent, így a Vonal menüpont halvány lesz, és egy pipa fog előtte állni. A megelőző példák alapján a DIESEL kifejezések a menüpont egyidejű letiltására és megjelölésére is felhasználhatók.

Hivatkozás egy legördülő vagy kurzor-almenüre

A korábban ismertetett hivatkozási konstrukcióhoz hasonlóan egy almenüt a `$Pn=xxx` kifejezéssel aktivizálhat vagy kapcsolhat ki. Ennek a `POPn` menüszakaszokra vonatkozó változatait ismertetjük ezen a helyen. A formátum:

```
$Pn.i=xxx
```

ahol

`$` az a speciális karakterkód, amely az AutoCAD rendszert egy menüszakasz betöltésére utasítja; a `Pn` a `POPn` menüterületet jelöli; `i` a menüpont száma; az *xxx* pedig, ha jelen van, a menüpont elhalványítására vagy megjelölésére szolgáló (előzőekben definiált) jeleket tartalmazó karakterláncot jelöli.

```
$P3.4=~
```

Elhalványítja a `POP3` menü 4. tételét.

```
$P7.1=!.
```

Pipa jellel látja el a `POP7` menü 1. tételét.

```
$P7.1=
```

Megszünteti a `POP7` menü 1. tételének elhalványítását és megjelölését.

A menüpontok számozása folyamatos, tekintet nélkül a menüfájl hierarchiájára; a címet követő legelső tétel az 1. tétel. Például:

```
***POP2
[Eszköz]                    Cím
[Súgó!]'?                   1. tétel
[Mégsem]^C^C^C              2. tétel
[--]                        3. tétel
[Vissza]^C^C_U              4. tétel
[Előre]^C^C_redo            5. tétel
[--]                        6. tétel
[->Traszter]                7. tétel
[Középpont]középp           8. tétel
```

Annak érdekében, hogy egy tétel egyszerűen hivatkozhasson önmagára — függetlenül a menü hierarchiájában elfoglalt helyétől — a következő kifejezésformák szintén megengedettek:

`$P@.@=xxx` Az aktuális vagy a legutóbb kiválasztott menüpontra hivatkozik.

`$P@.n=xxx` Az aktuális vagy a legutóbb kiválasztott menü *n*-edik tételére hivatkozik.

Hozzáférés a címkék státuszához AutoLISP és ADS függvényekkel

Az AutoLISP **menucmd** és az ADS **ads_menucmd()** függvényei elfogadják a `$P`*n*`=`*xxx* parancskifejezéseket, de a bevezető `$` jel nélkül. A kifejezések *xxx* részében két olyan érték szerepelhet, amelynek ezen függvények esetében speciális jelentése van.

`Pn.i=?` A megadott tétel elhalványítására, illetve megjelölésére vonatkozó aktuális státusszal mint karakterlánccal tér vissza. (Például "~" egy elhalványított, "!." egy kipipált és "" "" egy sem el nem halványított, sem meg nem jelölt tétel esetében.)

`Pn.i=#?` A fentivel azonos típusú karakterlánccal tér vissza, de a `Pn.`*i*`=` előtaggal együtt. Ez a `@` formákkal együtt hasznos, mivel a tényleges menüt és tételszámot adja vissza.

Például feltételezve, hogy a `POP6` menü 5. tétele pillanatnyilag halvány, a következő AutoLISP **menucmd** függvényhívás az itt látható értékekkel tér vissza:

```
(menucmd "P6.5=?")          eredménye "~"
(menucmd "P6.5=#?")         eredménye "P6.5=~"
```

További tudnivalók a II. az *AutoLISP Referencia című részben*, az "Az AutoLISP használata menükben" című résznél, a 126. oldalon.

Legördülő menük cseréje

Mivel az AutoCAD legördülő menüi lépcsőzetes láncolatot alkotnak, nincs igazán szükség a cserélgetésükre. A menük cseréje a felhasználói felület következetességét is megbonthatja. Ezen okokból nem javasoljuk, hogy ezt tegye. Ha azonban mindezek ellenére szükségesnek ítéli, a következőkben leírtak a hasznára lesznek.

A legördülő menük és almenüik a $ parancsok révén cserélhetők. A menük cseréje céljából a legördülő menüterületek P1-től P16-ig terjedő neveket viselnek. A menüsorban megjelenő cím nem más, mint a menüszakasz első sorában szereplő címke, így a címet ennek a sornak a $P*n*= paranccsal történő behelyettesítésével változtathatja meg. A $Pn=* speciális parancs hatására (bármely menüponton belül) az aktuálisan a POP*n* területhez tartozó menü jelenik meg (gördül le) a képernyőn .

Megjegyzés A legördülő menük cseréje nem kívánt eredménnyel járhat azon rendszerek esetében, amelyek "tear-off" ("leszakítható", azaz a képernyőn áthelyezhető) menüket alkalmaznak.

Megszorítások

Az alábbi parancsok alatt a menüsor és a legördülő, illetve a kurzormenük nem érhetők el:

- DSZÖVEG (miután a forgatási szöget beállította)
- SKICC (miután a rögzítési határértéket beállította)
- NÉZŐPONT (mialatt a koordinátarendszer és az iránytű látható)
- ZOOM Dinamikus
- DNÉZET

Ábrazóna-menük

Egy ábrazóna-menü a menüfájl ***IMAGE részének megadásával definiálható. Ez felülírja az addig használt ***ICON fájlrészt (a program ezen verziójához még érvényes a ***ICON név). Az ábra almenü a képernyő almenühöz hasonlóan definiálható, vagyis úgy, hogy minden tételhez egy címkét ad, ami után a tétel kiválasztásakor lefutó menümakró szövege áll. A POPn menürészekhez hasonlóan az almenü

első sora a cím. A címet a rendszer a menüt alkotó képek csoportja fölé írja ki, és az nem része a kiválasztható menüpontoknak.

Az ábrazóna-menü megadási módja

A $I= parancs vonatkozik az ábrazóna-menüre. Az almenüsorok között üres sort kell hagyni a tételeknek az előző almenütől való elkülönítésére.

Menüpont-címkék

Egy ábrazóna-menü címkéi általában a képernyőn megjelenített szöveges címkék helyett diafájl-nevekre vonatkoznak. A diafájl nevének - amely lehet egy önnálló dia vagy egy könyvtár része - ugyanolyannak kell lennie, mint amit egy HBEDIA parancsra gépelne be. A diafájl tartalmazza az adott választásra megjelenítendő képet.

Az ábrazóna-menü címkék egy görgethető listaablakban jelennek meg, amely maximum 17 karaktert tartalmazhat minden egyes címkéhez. A diafájl neve általában kiíródik, de az alábbi ábrazóna-menü címkézési módok is lehetségesek.

`[dianév]`

A listaablakban a *dianév* dia neve látható, ábraként a *dianév* nevű dia jelenik meg.

`[dianév, címkeszöveg]`

A listaablakban a *címkeszöveg* szöveg látható, ábraként a *dianév* nevű dia jelenik meg.

`[diakönyvt(dianév)]`

A listaablakban a *dianév* dia neve látható, ábraként a *diakönyvt* diakönyvtárban található *dianév* nevű dia jelenik meg.

`[diakönyvt(dianév,címkeszöveg)]`

A listaablakban a *címkeszöveg* szöveg látható, ábraként a *diakönyvt* diakönyvtárban található *dianév* nevű dia jelenik meg.

`[blank]`

Ha a menüpont címkéjeként a *blank* szöveget adja meg, akkor a listaablakban egy tagolóvonal lesz látható, az ábrak között pedig egy üres ábra jelenik meg.

`[címkeszöveg]`

Amennyiben a menüpont címkéje szóközzel kezdődik, akkor a listaablakban a *címkeszöveg* szöveg lesz látható, ábra pedig nem jelenik meg. Ez lehetővé teszi parancsok hozzárendelését olyan

egyszerű menüpontokhoz, mint például a Kilép, anélkül, hogy ezeket a szavakat diába kellene foglalni.

Az ábrazóna-menü megjelenítése

A speciális `$I=*` menüparancs minden rendszer esetében az aktuális ábrazóna-menüt jeleníti meg, és lehetővé teszi, hogy abból válasszon. Ezt a parancsot bármely menüszakaszban elhelyezheti, de a billentyűzetről nem adhatja ki. Az ábrazóna-menü egy párbeszéd-panelben jelenik meg, melynek címe — felül, középre igazítva — az ábrazóna-menü első sora lesz. Az ábrazóna-menüvel együtt az AutoCAD egy nyíl alakú kurzort is megjelenít. Amikor a szöveges címkék egyikét kiválasztja, az AutoCAD ezt azzal nyugtázza, hogy a megfelelő ábra körül egy keretet rajzol. Hasonlóképpen, egy ábra kiválasztásakor a hozzá tartozó szöveg jelenik meg kiemelten. Ha dupla kattintással e tételek bármelyikére rámutat, vagy kiválasztja az OK nyomógombot, az AutoCAD végrehajtja az adott ábrához tartozó menüparancsokat.

Az ábrazóna-menüből úgy léphet ki, ha egy tételét kiválasztja, vagy megnyomja a Ctrl+C billentyűket; minden egyéb billentyű hatástalan. Alapértelmezés szerint az OK és a Mégsem nyomógomb áll a rendelkezésére.

Egy tétel kiválasztása az ábrazóna-menüből

Amikor az ábrazóna-menüből kiválaszt egy menüpontot, a menü sorában megadott szöveg logikailag a menüt aktivizáló `$I=*` parancs helyére kerül. Ezáltal olyan kevés, illetve olyan sok parancsot adhat meg, amennyit az alkalmazás szükségessé tesz. Ez ugyanakkor azt is jelenti, hogy az ábrazóna-menüben nem élhet a menüpont ismétlésének a lehetőségével. Az ábrazóna-menü tételének kiválasztásával megadott szöveget a menükezelő azonnal beolvassa. Ez a szöveg menüparancsokat — beleértve a `$I=` parancsokat — is tartalmazhat. Így hierarchikus felépítésű ábrazóna-menük készítésére is van lehetőség; ekkor egy tétel kiválasztása egy másik ábrazóna-menüt jelenít meg, és így tovább. Mivel e menük aktivizálása szekvenciális, nem pedig egymásba ágyazott, a készíteni kívánt struktúra bonyolultságának legfeljebb csak az érthetőség és a követhetőség szab határt.

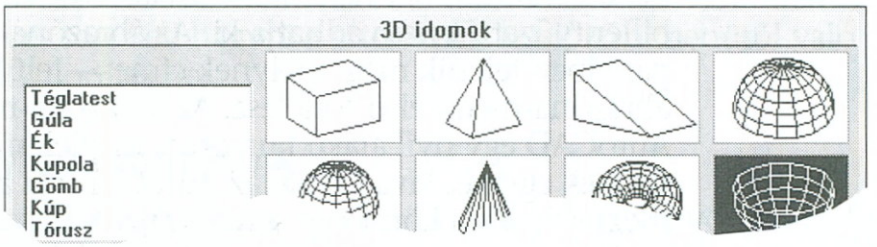
Minták az ábrazóna-menüre

A következő példa a `**POLY` nevü ábraalmenüt mutatja be:

```
**ábra_3DObjektumok
[3D Objektumok]
[acad(Doboz3d,3D Doboz)]^c^cai_doboz
[acad(Gúla)]^c^cai_gúla
[acad(Ék)]^c^cai_ék
[acad(Félgömb)]^c^cai_dome
[acad(Gömb)]^c^cai_gömb
```

```
[acad(Kúp)]^c^cai_kúp
[acad(Tórusz)]^c^cai_tórusz
[acad(Korong)]^c^cai_korong
[acad(Síkháló)]^c^cai_síkháló
```

valamint az ennek eredményeképpen megjelenő ábrazóna-menü részletét:



A **POLY nevű ábrazóna-almenü

Az alábbi példa egy elektronikus alkatrészek beillesztésére szolgáló ábrazóna-menüt mutat be. Figyelje meg annak a szöveges címkének a használatát, amely a rögzítőelemeket tartalmazó másik almenüre vált át.

```
***ÁBRA
**IRÉSZEK
[Elektronikus alkatrészek]
[kond]^Cbeill kond
[ell]^Cbeill ell
[neon]^Cbeill neon
[trioda]^Cbeill trioda
[tetroda]^Cbeill tetroda
[ Rögzítők]$I=irögz $I=*

**IRÖGZ
[Rögzítők]
[csavar632]...
```

Ennek az ábrazóna-menünek az aktivizálásához az alábbihoz hasonló tételt kell kiválasztania bármely menüből:

```
[Elektronikus alkatrészek]$I=irészek $I=*
```

Az alábbi változatban az ábrák egy *eknyvt* diakönyvtárból származnak (a listaablakban csak a tényleges dianevek jelennek meg).

```
***ÁBRA
**IRÉSZEK
[Elektronikus alkatrészek]
[eknyvt(kond)]^Cbeill kond
[eknyvt(ell)]^Cbeill ell
[eknyvt(neon)]^Cbeill neon
[eknyvt(trioda)]^Cbeill trioda
```

```
[eknyvt(tetroda)]^Cbeill tetroda
```

Ha az egymást követő ábrákat az AutoCAD ugyanabból a diakönyvtárból jeleníti meg, akkor a könyvtárfájl nyitva marad. Ez jelentősen lerövidíti az ábrazóna-menü megjelenítéséhez szükséges időt.

Az ábrazóna-menük diáinak elkészítése

Bármely, az AutoCAD által előállított diát felhasználhatja ábraként. Ugyanakkor az ábrazóna-menük optimális alkalmazásához néhány dolgot figyelembe kell vennie az ábrák céljára szolgáló diaképek elkészítésekor. Az alábbiakban a diaképek elkészítéséhez adunk irányelveket:

- Tartsa szem előtt az egyszerűséget. Az ábrazóna-menü megjelenítésekor a felhasználónak ki kell várnia, amíg az AutoCAD az összes ábrát kirajzolja a képernyőre, és csak azután választhat ki egy menüpontot. Amennyiben az ábrazóna-menüben bonyolult szimbólumokat kíván bemutatni, érdemes a teljes részletesség helyett a szimbólumok egyszerűsített változatát használni ábraként. Az ábra ne legyen bonyolultabb annál, mint amennyi az azonnali és egyértelmű azonosításhoz szükséges.
- Töltse ki a mezőt. Gazdálkodjon jól a képernyőn rendelkezésre álló területtel, és ne felejtse el, hogy az ábrák a teljes képernyőnek csak egy kis részletén fognak megjelenni. Amikor az ábrához elkészíti a diát, ügyeljen arra, hogy az ábra teljesen kitöltse a képernyőt, mielőtt kiadja a KIDIA parancsot. Nagyon széles és rövid, illetve hosszú és keskeny ábra esetén a legesztétikusabb képet akkor kapja, ha a dia elkészítése előtt az ábrát a TOL paranccsal középre állítja.

  Az ábrák 1.5:1 arányú (1.5 egység széles és 1 egység magas) képek. Ha az Ön grafikus képernyőjének arányai ettől eltérőek, nehéz olyan diákat készíteni, amelyek az ábrazóna-menüben a mező közepén helyezkednek el. Ha azonban egy olyan nézetablakban készíti el a diát, amelynek arányai éppen megfelelők (1.5:1), akkor biztos lehet benne, hogy az ábra az ábrazóna-menüben is azonos helyzetben fog megjelenni.

A következő parancssorozat egy megfelelő arányokkal rendelkező nézetablakot hoz létre (olyan rajzot feltételezve, amelyben még nincsenek nézetablakok, és a TILEMODE rendszerváltozó értéke 1):

Parancs: **tilemode**
TILEMODE új értéke <1>: **0**
A modelltérbeli nézetablakok beillesztésére használja az MNÉZET parancsot.
Parancs: **MNÉZET**
BE/KI/Takartrajz/Illeszt/2/3/4/Visszaállít/<Első pont>: **0,0**
Másik sarokpont: **3,2**
Parancs: **zoom**
Mind/Közép/Dinamikus/Terjedelem/Bal/Előző/Vmax/Ablak /<Arány(X/XP)>: **e**
Parancs: **mtér**

Ez a nézetablak már az ábra arányaival rendelkezik.

- A kitöltött területek, mint a nagy kiterjedésű vonalláncok, vastag vonalak, vagy a kitöltött testek csak akkor jelennek meg egy ábratételben, ha a dia a KITÖLT parancs használata után generálódott.
- Ne tévessze szem elől a végcélt. Az ábrák elsődleges rendeltetése, hogy a felhasználó grafikus szimbólumok közül választhasson.

Táblamenük

Az AutoCAD a digitalizáló táblán legfeljebb négy, parancsok bevitelére használható menüterület konfigurálását teszi lehetővé. Az ezekre a területekre vonatkozó menümakrók a menüfájl TABLET1-4 címkékkel jelölt szakaszaiban definiálhatók.

A TABLETn menüszakaszok tételeinek a megadási módja megegyezik a többi szakaszéval. A tételek címkéit — a BUTTONSn szakaszhoz hasonlóan — az AutoCAD megjegyzésnek tekinti, és nem jeleníti meg a képernyőn.

A TÁBLA Kfg paranccsal definiált menüterületek egyenlő méretű mezőkre oszlanak, melyek a menüből történő választásra szolgálnak; méretüket az egyes menüterületeken definiált sorok és oszlopok száma szabja meg. A TABLETn szakaszcímkét követő sorok közvetlenül egy-egy mezőnek felelnek meg, balról jobbra és felülről lefelé haladva a táblán (függetlenül attól, hogy egy sor tartalmaz-e szöveget vagy sem).

Abban az esetben például, ha Ön egy 5 oszlopból és 4 sorból álló menüterületet konfigurál, a fájlban a szakaszcímkét közvetlenül követő sor a táblán a legfelső sor bal szélső mezőjének felel meg. Hasonlóképpen, a fájlban a szakaszcímkét követő nyolcadik sor a

táblán a második sorban, balról a harmadik mezőhöz tartozik. Az AutoCAD a táblamenüre vonatkozóan szakaszonként legfeljebb 32 766 menüpontot fogad el, amely valószínűleg több mint elegendő bármely táblamenü elkészítéséhez.

Saját menümakróit az *acad.mnu* `***TABLET1` szakaszában helyezheti el. Ezen a területen belül a menüpontok a táblasablon legfelső 225 mezőjének felelnek meg (a sorok A-tól I-ig, az oszlopok 1-től 25-ig terjednek). Azokat a sorokat veheti igénybe a saját céljaira, amelyekben csak a címke található.

Keresse meg a menüfájl `***TABLET1` sorát, és figyelje meg, hogy az azt követő 225 sor az alábbi címkéket tartalmazza:

```
***TABLET1
[A-1]
[A-2]
[A-3]
.
.
.
[I-25]
```

Ezek a címkék a sablon hálórendszerének felelnek meg. Saját menümakróit a megfelelő [*sor-oszlop*] címke sorába írhatja be, az ebben a fejezetben korábban ismertetett formátumban. Az `[I-25]` mezőt követő sorok megváltoztatását nem javasoljuk.

Az AutoLISP használata menükben

Az összetett feladatot elvégző menüpontok kialakításához — az *AutoLISP Referenciában* ismertetett — AutoLISP változókat és kifejezéseket is felhasználhatja.

Például egy olyan alkalmazásban, amely blokkok beillesztéséhez előre beállított változókat használ, az alábbiakhoz hasonló menüpontok szerepelhetnek:

```
[Beállít:]
[ABLSZÉL]^C^C(setq ABLSZÉL (getreal "\nAdja meg az ablak szélességét:
"))
[FALMAGS]^C^C(setq FALMAGS (getreal "\nAdja meg a fal magasságát: "))

[Ablak   ]^C^CBEILL ablak Xméretarány !ABLSZÉL Yméretarány !FALMAGS
```

Ez adott esetben beilleszti az "ablak" blokkot úgy, hogy *X* irányban az aktuális ablakszélességnek, *Y* irányban pedig az aktuális falmagasságnak megfelelően állítja be a méretarány-tényezőt. A példában az aktuális értékeket a felhasználó által definiált ABLSZÉL és FALMAGS szimbólumok tárolják. A forgatási szög meghatározása a felhasználóra van bízva, így az ablak elforgatható a falban.

A következő menüpontok a fogópontok méretének menet közbeni módosítását teszik lehetővé:

```
[GRIP-
fel]^P(setvar"gripsize"(1+(getvar"gripsize")))(redraw)(princ)
[GRIP-le ]^P(setvar"gripsize"(1-(getvar"gripsize")))(redraw)(princ)
```

Ezeket a menüpontokat továbbfejlesztheti úgy, hogy azok ellenőrizzék az értékeket, továbbá megakadályozzák a 0-nál kisebb, illetve 255-nél nagyobb értékek átadását a GRIPSIZE rendszerváltozónak.

Az alábbi menüpont két pont megadását kéri a felhasználótól. Ezt követően egy téglalap alakú vonalláncot rajzol, ahol a megadott pontok a sarokpontokat jelölik.

```
[DOBOZ](setq a (getpoint "Első sarokpont: "));\+
(setq b (getpoint "Átellenes sarokpont: "));\+
vlánc !a (list (car a)(cadr b)) !b (list (car b)(cadr a)) c;
```

Ez már az AutoCAD menü adaptálási lehetőségeinek magasszintű kihasználása. Ha hasonló menüpontokat szeretne készíteni, tanulmányozza figyelmesen ezt a példát, valamint a *II. "AutoLISP Referencia" című részében* található információkat. A kísérletezés és a gyakorlat segítségére lesz abban, hogy hatékonyan használja ezt a szolgáltatást.

Megjegyzés Az AutoCAD legfeljebb 255 karakter hosszú AutoLISP kifejezést fogad el a menüfájlban. Ha hosszabbra van szüksége,

szakítsa meg a kifejezést egy pontosvesszővel, hogy azt az AutoCAD több részben olvashassa be és hajthassa végre.

Arra vonatkozóan, hogy az AutoLISP segítségével hogyan hívhat meg egy párbeszédpanelt a menüfájlból, lapozza át a 10., "Párbeszédpanelek kezelése" című fejezetet.

Az AutoLISP kifejezések menümakrókban történő felhasználásának kényelmes és egyben hatékony módja, ha az AutoLISP utasításokat egy különálló *.mnl* fájlban helyezi el, megkönnyítve ezáltal a menüfájl és az AutoLISP program elkészítését és karbantartását is. Az AutoCAD a menüfájl betöltésekor automatikusan betölti a vele megegyező nevű *.mnl* fájlt.

Windows-specifikus menüjellemzők

Az AutoCAD Windows alatti verziója más platformokon el nem érhető kiegészítő menü szolgáltatásokat kínál.

Segítségével az alapmenüvel együtt részleges menüfájlokat is betölthet. Ez lehetővé teszi, hogy a felhasználók és a fejlesztők kihasználják a többszörös menüket. Ezen kívül az AutoCAD Windows változata kényelmes módszert kínál a különböző menüpontokra való hivatkozásra. Minden menüponthoz egy névcímke, és minden menüfájlhoz egy menü csoportnév tartozik. A menü csoportnév és a névcímke megadásával bármely menüpont be- és kikapcsolható, vagy annak állapota ellenőrizhető. A névcímke-jelölés lehetővé teszi az egyes menüpontokra való hivatkozást azok betöltődése alatt. Ez a módszer érzéketlen a menü sorrendjének dinamikus változásra.

A 90. oldalon a „Menüszakasz-címkék" című táblázatban a felsoroltakon kívül az alábbi táblázatban található menürészek használhatók menüfájlokban.

Menüszakasz-cimkék

Részcímke	*Leírás*
***MENUGROUP=	A menüfájl csoportneve
***TOOLBARS	Eszköztár definíciók
***HELPSTRINGS	
***ACCELERATORS	Gyorsító billentyű definíciók

Névcímkék és menücsoportok

A ***MENUGROUP= címkének meg kell előznie minden, a névcímke módszert használó menürész-definíciót. Ez a címke adja meg a menücsoport nevét minden további menü-névcímke definíció számára. Egy MENUGROUP definíció egy maximum 32 alfanumerikus karakterből álló karakterlánc (szóközök és írásjelek nem lehetnek benne).

A **menucmd** AutoLISP függvény menüpont-címkékre való hivatkozásra és azok módosítására szolgál. Mivel a Windows menüfájlok részleges menük betöltése folytán dinamikusak, az alábbi (DOS és UNIX menük által használt) megadási mód működése nem minden esetben garantált.

```
[Tiltott vonal régi megadás](menucmd "P1.2=~")
```

Ez a megadási mód a menüpont helyzetén alapul és nem működik, ha a menü írója egy új tételt szúr be a POP1 részbe, vagy ha a MENÜBETÖLT parancs egy új legördülő menüt szúr be a POP1 rész elé.

Mivel minden menünek megvan a saját menücsoport-címkéje, a többszörös részleges menük használhatják ugyanazt a címkét egy névcímkében. A névcímke ezért csak abban a menüfájlban szerepelhet, amelyben definiálva van. Ez lehetővé teszi, hogy a részleges menüfájlokat betöltő alkalmazások elérhessék egymás tételeit.

A névcímke egy alfanumerikus karaktereket tartalmazó karaktersorozat (az aláhúzás karakter megengedett), amely közvetlenül megelőzi a menüpont-címkét. A *menucmd* függvény segítségével a névcímkére való hivatkozáshoz használja a következő megadási módot.

```
(menucmd "Gmenücsoport.névcímke=érték")
```

Az alábbi menükód az előző példával megegyező feladatot lát el azzal a különbséggel, hogy ez a menüpontnak a menüben elfoglalt helyétől függetlenül működik.

```
[Tiltott vonal](menucmd "Gminta.ID_Line=~")
```

Az alábbi részleges menü bemutatja a névcímke és a menücsoport jellemzőit.

```
// Ez egy példa a részleges menüre.
//Bemutatja a menü névcímke megadási módjának alapvető, új
működését.
***MENUGROUP=sample
***POP1
**AnAlias
ID_Title  [/CCím]
ID_Cancel [Mégsem parancs]^C^C
ID_Line   [/VVonal]^C^C_line
```

```
            [Tiltott vonal](menucmd "Gminta.ID_Line=~")
            [Ellenőrző vonal](menucmd "Gminta.ID_Line=!.")

***POP2
[/2Title2]
[Másik legördülő](menucmd "Gminta.ID_Line=~")
```

Részleges menük

A részleges menük a MENÜBETÖLT parancs segítségével tölthetőek be egy már létező alapmenübe (pl. *acad.mnu*). Az *alapmenü* kifejezés egy teljes menüleírást jelent, és általában nyomógomb-, képernyő-, kép- és egyéb részeket tartalmaz a POPn részben található legördülő menüdefiníciókon kívül. Csakúgy, mint az alap menüfájlok esetében, ha a részleges menüfájl nevével megegyező nevű kapcsolódó *.mnl* fájl létezik, akkor az abban levő rutinok töltődnek be. Ugyanígy, ha létezik egy kapcsolódó *.dll* menü forrásfájl, akkor az abban levő rutinok töltődnek be.

Részleges menük betöltése

A felhasználó a MENÜBETÖLT Menüadaptációs párbeszédpanel segítségével módosíthatja a menüsort és tölthet be részleges menüket. Ahhoz, hogy egy alkalmazás betölthessen egy részleges menüt, a MENÜBETÖLT parancs parancssori alakját kell használnia, amit megtehet úgy, hogy a FILEDIA rendszerváltozót 0-ra állítja, a majd kiadja a parancsot. Az alábbi példa az AutoLISP **setvar** függvényét használja a FILEDIA 0-ra állítására majd ezután; a **command** függvényt a MENÜBETÖLT parancs kiadására, és a *testmenu* nevű fájlt adja meg betöltendő részleges menüfájlként.

Parancs: **(setvar "filedia" 0)**
Parancs: **(command "menübetölt" "testmenu")**

Hasznos, ha egy alkalmazás visszaállítja a rendszerváltozókat (mint például a FILEDIA) eredeti állapotukba, ha használatukat az adott alkalmazás tovább már nem igényli.

Egy menü betöltése után az alkalmazásnak minden újonnan definiált legördülő menüt el kell helyeznie a **menucmd** függvény segítségével, az alábbi példákon látható módon:

Parancs: **(menucmd "P6=+sample.pop1")**
nil
Parancs: **(menucmd "P7=+sample.pop2")**
nil

Ez a megadási mód hasonlít a "Hivatkozás egy legördülő vagy kurzormenüre" című résznél, a 117. oldalon tárgyaltakra, azzal a kiegészítéssel, hogy a kifejezés bal oldalán azon legördülő menü neve szerepel, amely elé az új menüt akarjuk beilleszteni. Ha egy új menüt

kíván a menühöz illeszteni, használjon egy olyan számot, amely nagyobb a pillanatnyilag kijelzett legördülő menük számánál. A kifejezés jobb oldalán egy + (pluszjel), a betöltődött menücsoport neve (pl. MENÜCSOPORT=minta), egy pont, majd a menü álneve szerepel. Ebben a példában az első legördülő menünek két álneve volt definiálva: a POP1 és az AnAlias. A pontot követő paraméterként ezek közül bármelyiket használhatnánk ebben a példában.

Ha a részleges menü írója ismeri az alapmenü tartalmát, akkor a menüpont megadási módjában hivatkozhat az alapfájlból származó címkére. Az *acad.mnu* nevű alapfájl egy részlete a következőképpen is alakulhat:

```
***MENÜCSOPORT=ACAD
***POP0          (és így tovább....)
 ....
***POP6
[Help]
ID_Contents [Tartalom]^C^C_HELP
ID_About    [Névjegy]^C^C_ABOUT
```

A részleges menü második legördülő menüje - a *sample.mnu* - módosítható úgy, hogy tartalmazzon egy olyan menüpontot, amely az alapmenüben levő címkére hivatkozik.

```
***POP2
[/2Cím2]
[Másik legördülő](menucmd "Gminta.ID_Line=~")
[Tiltott súgótartalom](menucmd "GAcad.ID_Contents=~")
```

Ezzel a módszerrel a többszörös részleges menüfájlok és a különböző alapfájlok együtt tudnak működni. Az AutoCAD szigorú MENÜCSOPORT-definíciót használ azért, hogy két menü ne adhassa meg ugyanazt a MENÜCSOPORT nevet. Egy a MENÜCSOPORT névvel konfliktusban levő menü betöltésének kísérlete a MENÜCSOPORT parancs törlését eredményezi.

Részleges menük törlése

A MENÜTÖRÖL parancs törli a részleges menüt az alapmenüből, így eltünteti a legördülő menüket a menüsorból. Lehetőség van ezen kívül arra is, hogy a legördülő menüket a teljes menücsoport törlése nélkül távolítsuk el. A megadási mód hasonló a legördülő menük megjelenítésekor használthoz:

Parancs: **(menucmd "P6=–")**
nil
Parancs: **(menucmd "P7=–")**
nil

Ez a kód eltünteti a képernyőről az aktuális menü 6. és 7. helyén levő legördülő menüket . A menücsoport továbbra is be van töltve, de a felhasználó egyik menüt sem látja.

A felhasználói menüsor tartalmának túl sűrű változásai a kezelői felület összezavarodását eredményezhetik. A menüsor állapotának látható megváltoztatása csak a felhasználó kifejezett kérése esetén ajánlatos. Például, ha a felhasználó törölni akar egy alkalmazást, akkor azok a menük, amelyekre csak ez az alkalmazás hivatkozott, szintén törölhetők.

Lehetséges ezen kívül a menünek a MENÜ parancs segítségével való teljes újrainicializálása, minden jelenleg betöltött részleges menü törlésével és egy új alapmenü fájl betöltésével. Ez az eljárás törli az összes részleges menüt, azok kapcsolódó címkedefinícióival együtt.

Eszköztárak

Az ESZKÖZTÁRAK rész meghatározza az alapértelmezésbeli elrendezést és az eszköztárak tartalmát. A `***ESZKÖZTÁRAK` részben a menüben definiált minden eszköztárhoz egy tartozik egy almenü. Az alábbi példában a `**ESZKÖZÖK1` deklaráció egy olyan almenü, amely a `ESZKÖZÖK1` nevű álnevet használja a következő eszköztár-definícióra hivatkozó címkeként.

Az eszköztárak számára meghatározott tételeknek ötféle típusa létezik. Az alábbi példa bemutatja minden egyes típus megadási módját. Az elválasztáson kívül minden sor egy szabványos névcímkével kezdődik, amelynek segítségével a tételhez súgó információ rendelhető.

```
***ESZKÖZTÁRAK
**ESZKÖZÖK1
CÍMKE1 [Eszköztár ("eSornév", tájolás, látható, xért, yért, sorok)]
CÍMKE2 [Nyomóomb ("gmbnév", id_small, id_large)]makro
CÍMKE3 [Leporelló ("flyname", id_small, id_large, ábra, álnév)]makro
TAG4 [Vezérlés (elem)]
[--]
```

Az eszköztár-definíció első sora (`TAG1`) meghatározza az eszköztár-definíció jellemzőit. Ebben a `Eszköztár` kulcsszót zárójelben levő opciók sorozata követi. Az opciók az alábbiak:

esornév Az eszköztárat megnevező karakterlánc. Alfanumerikus karakterekből kell állnia, és nem lehet benne más elválasztójel, csak vonal (-) és aláhúzás (_).

Ez a név az álnévvel együtt lehetővé teszi az eszköztárra való programozott utalást.

tájol A tájolási kulcsszó. A lehetséges értékek `Lebegő`, `Fent`, `Lent`, `Bal` és `Jobb`. Írhatóak nagy-, vagy kisbetűkkel.

látható A láthatóság kulcsszava. A lehetséges értékek `Mutat` és `Takar`, lehetnek nagy-, vagy kisbetűk.

xért Pixelekben adja meg az X koordinátát, amely a képernyő bal széle és az eszköztár jobb oldala között mérhető.

yért Pixelekben adja meg az X koordinátát, amely a képernyő felső széle és az eszköztár felső oldala között mérhető.

sorok A sorok számát meghatározó számérték.

A második sor (`CÍMKE2`) egy nyomógombot definiál. Ebben a `nyomógomb` kulcsszót zárójelben levő opciók sorozata követ. Az alábbi opciók lehetségesek:

nygnév A nyomógombot megnevező karakterlánc. Alfanumerikus karakterekből kell állnia, és nem lehet benne más elválasztójel, csak vonal (-) és aláhúzás (_). Ez a karakterlánc jelenik meg a képernyőn Eszköztippként, amikor a kurzor a nyomógomb fölé kerül.

azon_kicsi Karakterlánc, amely a kisméretű (16X16-os bittérkép) forráskép azonosító karakterláncának nevét tartalmazza. Alfanumerikus karakterekből kell állnia, és nem lehet benne más elválasztójel, csak vonal (-) és aláhúzás (_). Egy felhasználó által definiált bittérképet is meghatározhat.

id_nagy Karakterlánc, ami a nagyméretű (32X32-es bittérkép) forráskép azonosító karakterláncának nevét tartalmazza. Alfanumerikus karakterekből kell állnia és nem lehet benne más elválasztójel, csak vonal (-) és aláhúzás (_). Egy felhasználó által definiált bittérképet is meghatározhat.

makró A definíció után egy parancs-karakterlánc áll, ami megfelel a parancs-karakterláncokra vonatkozó szabványos menüpont megadási módnak.

A harmadik sor (`CÍMKE3`) egy leporelló definíciót tartalmaz. Ebben a `Leporelló` kulcsszót zárójelben levő opciók sorozata követi. Az alábbi opciók lehetségesek:

lepnév A leporellót megnevező karakterlánc. Alfanumerikus karakterekből kell állnia, és nem lehet benne más elválasztójel, csak vonal (-) és aláhúzás (_). Ez a karakterlánc jelenik meg a képernyőn Eszköztippként, amikor a kurzor a leporelló fölé kerül.

azon_kicsi Karakterlánc, amely a kisméretű (16X16-os bittérkép) forráskép azonosító karakterláncának nevét tartalmazza. Alfanumerikus karakterekből kell állnia, és nem lehet benne más elválasztójel, csak vonal (-) és aláhúzás (_). Egy felhasználó által definiált bittérképet is meghatározhat.

azon_nagy Karakterlánc, ami a nagyméretű (32X32-es bittérkép) forráskép azonosító karakterláncának nevét tartalmazza. Alfanumerikus karakterekből kell állnia, és nem lehet benne más elválasztójel, csak vonal (-) és aláhúzás (_). Egy felhasználó által definiált bittérképet is meghatározhat.

ábra Boolean kulcsszó, ami azt szabályozza, hogy mindig a saját ábrája, vagy az utoljára kiválasztott (másik) ábra jelenjen-e meg. A lehetséges értékek `Sajátábra` és `Másábra`; lehetnek nagy-, vagy kisbetűsek.

álnév A leporellóként megjelenítendõ eszköztárra való hivatkozás. Az álnév egy a szabványos *`**álnév`* megadási móddal meghatározott eszköztár-almenüre utal.

makró A definíció után egy parancs-karakterlánc áll, ami megfelel a parancs-karakterláncokra vonatkozó szabványos menüpont megadási módnak.

A negyedik sor (`CÍMKE4`) egy speciális vezérlőegységet definiál. Itt a `Vezérlõ` kulcsszó után a hivatkozott vezérlőelem típusát meghatározó név áll, zárójelek között.

elem Ennek a paraméternek három lehetséges értéke van. Ezek írhatóak nagy- vagy kisbetűkkel.

A `Fólia` opció a fólia vezérlőegységet határozza meg. Ez az elem egy legördülő listaablak, melynek segítségével a rajz pillanatnyi fóliái szabályozhatók.

A `Vonaltípus` opció a vonaltípust vezérlő elemet határozza meg. Ez az egység egy legördülő listaablak,

melynek segítségével az aktuális vonaltípus jellemzői szabályozhatóak.

A `Szín` opció a színvezérlő egységet határozza meg. Ez egy nyomógomb, amely a DDSZIN parancsot adja ki.

Az ötödik sor egy elválasztót definiál (--).

A felhasználó által definiált bittérképeket az azon_kicsi és azon_nagy forráskép-nevek helyén lehet használni. A felhasználó által definiált bittérképnek megfelelő méretűnek kell lennie (16 pixeles négyzetnek az `id_kicsi`, és 32 pixelesnek az `id_nagy` paraméter esetén), és a lemezen a Support könyvtárban kell lennie. A felhasználó által definiált bittérképet az alábbi példán látható módon a fájlnévvel és a *.bmp* kiterjesztéssel kell megadni.

```
címke34 [Nyomógomb ("Az én parancsom", mycmd16.bmp,
mycmd32.bmp)]^C^CMYCMD
```

Menü-specifikus súgó

A modern grafikus felülettel most az AutoCAD beépített része az on-line súgó is. Az on-line rendszerszintű súgóra és az állapotsori súgóra vonatkozó index is elérhető.

Menüpontok súgója

Az on-line rendszerszintű súgó (Windows súgófájlok) eléréséhez elegendő, ha egy menüpontot megfeleltet egy indexszel és egy csatolt, külső súgófájllal. A menücímkét egy kapcsolódó súgófájl indexeként használja.

Ha a felhasználó meghívja a rendszerszintű súgót, (az F1 billentyű megnyomásával, amikor a kívánt menüpont van kijelölve) akkor az két argumentum alapján töltődik be. Az egyik az eredeti menüfájl neve, a másik a hozzá tartozó menü címkéje. Például az `ID_Cancel` névcímkével rendelkező menüpont az `ID_Cancel` értékével és *sample.hlp* súgófájllal hívná meg a súgót.

Állapotsor-súgó—Súgó karakterláncok

A beépített súgórendszer másik részét az állapotsor-súgó üzenetei képezik. Ezek olyan egyszerű, magyarázó szövegek, amelyek egy menüpont kiválasztása esetén, az állapotsorban jelennek meg. A SÚGÓKARAKTERLÁNC menürész a a következő súgóformátumot támogatja.

```
//Ez ugyanaz a példa, amit az előzőekben, minden
//menücímkéhez hozzáillesztett súgóindexekkel
//kapcsolatban használtunk.
//Ez az állaptsor-súgót támogató ***SÚGÓKARAKTERLÁNC
//szakasz használatához lett kiterjesztve.
```

```
*** MENUGROUP=sample
***POP1
ID_Title   [/TTitle]
ID_Cancel  [Mégsem parancs]^C^C
ID_Line    [/VVonal]^C^C_line
           [Tíltott vonal](menucmd "Gminta.ID_Line=~")
           [Ellenőrző vonal](menucmd "Gminta.ID_Line=!.")

***POP2
[/2Title2]
[Másik legördülő](menucmd "Gsample.ID_Line=~")

***HELPSTRINGS
ID_Title   [Ez a Cím menü]
ID_Cancel  [Ez a tétel törli az előző parancsot]
ID_Line    [Ez egy egyszerű vonalat rajzol]
```

A SÚGÓKARAKTERLÁNC rész megadási módja egy névcímkéből, és az azt követő címkéből áll. Ha a menüpont ki van választva, akkor a menücímke lekérdezi a SÚGÓKARAKTERLÁNC részben a megfelelő bevitelt. Ha megtalálja a keresett részt, a címkében levő karakterlánc megjelenik az állapotsorban.

Gyorsító billentyűk

Az AutoCAD Windows alatti verziója támogatja a felhasználó által definiált gyorsító billentyűket. A következőkben egy rövid példát mutatunk be.

```
***ACCELERATORS
ID_Line     [SHIFT+CONTROL+"L"]
ID_Cancel   ["ESCAPE"]
            [CONTROL"Q"]^C^C_quit
```

A GYORSITÓK szakasz olyan tételeket tartalmaz, amelyek két formátumot tárolnak egyben. Az első a névcímke (mint az `azon_Vonal`), amit egy módosítókat tartalmazó címke követ. A módosítókat vagy egy egyszerű karakter, vagy egy speciális billentyű követi (például az `"ESCAPE"`), idézőjelek között. A tételek ezen típusa egy billentyűsorozatot rendel a menüponthoz. A plusz jel (+) segítségével több módosító összefűzésére is lehetőség van, ahogyan az első példa is mutatja. Ha a rendszer egy különleges billentyűsorozatot észlel, akkor a névcímkéhez rendelt menüpont végrehajtódik, mintha a felhasználó kiválasztotta volna a menüpontot.

A következő táblázat az érvényes módosítók listáját tartalmazza.

Érvényes módosítók

Karakterlánc	*Leírás*
CONTROL	A Ctrl billentyű (a legtöbb billentyűzeten megtalálható).
SHIFT	A Shift billentyű (a jobb- vagy a baloldali).
COMMAND	Az Apple ("alma") billentyű a Macintosh billentyűzeteken.
META	A UNIX alapú gépek billentyűzetén található meta billentyű.

A következő táblázat a virtuális billentyűket sorolja fel. (Ezeket idézőjelek közé kell tenni.)

Virtuális billentyűk

Karakterlánc	*Leírás*	*Karakterlánc*	*Leírás*
"F1"	F1 billentyű	"NUMPAD0"	0 billentyű
"F2"	F2 billentyű	"NUMPAD1"	1 billentyű
"F3"	F3 billentyű	"NUMPAD2"	2 billentyű
"F4"	F4 billentyű	"NUMPAD3"	3 billentyű
"F5"	F5 billentyű	"NUMPAD4"	4 billentyű
"F6"	F6 billentyű	"NUMPAD5"	5 billentyű
"F7"	F7 billentyű	"NUMPAD6"	6 billentyű
"F8"	F8 billentyű	"NUMPAD7"	7 billentyű
"F9"	F9 billentyű	"NUMPAD8"	8 billentyű
"F10"	F10 billentyű	"NUMPAD9"	9 billentyű
"F11"	F11 billentyű	"FEL"	3 billentyű

Virtuális billentyűk (*folytatás*)

Karakterlánc	*Leírás*	*Karakterlánc*	*Leírás*
"F12"	F12 billentyű	"LE"	4 billentyű
"HOME"	Home billentyű	"BAL"	1 billentyű
"END"	End billentyű	"JOBB"	2 billentyű
"INSERT"	Ins billentyű	"ESCAPE"	Esc billentyű
"DELETE"	Del billentyű		

5

AZ ÁLLAPOTSOR KONFIGURÁLÁSA ÉS A DIESEL KARAKTERES KIFEJEZÉSNYELV

A DIESEL (Direct Interpretively Evaluated String Expression Language — közvetlen, értelmező programmal kiértékelt karakteres kifejezésnyelv) nyelv a MODEMACRO rendszerváltozó segítségével felhasználható az AutoCAD állapotsorának megváltoztatására. Menüpontokban is használható makrónyelvként, az AutoLISP helyett. A DIESEL kifejezések bemenő adatai és eredményei is karakterláncok.

Mivel a DIESEL kifejezések kizárólag karakterláncokkal dolgoznak, a USERS1-5 rendszerváltozók jól használhatók arra, hogy információt adjunk át egy AutoLISP rutinból egy DIESEL kifejezésnek. Az AutoLISP rutinok a DIESEL kifejezéseket az AutoLISP **menucmd** függvénye segítségével értékelik ki.

Az állapotsor konfigurálása—MODEMACRO

Az üzemmódokat jelző állapotsor igen értékes eszköz. Közvetlenül visszajelzi az aktuálisan bekapcsolt üzemmódokat, így egy pillantásra látható, hogy szükség van-e üzemmódváltásra a következő művelet végrehajtása előtt. DOS esetében az állapotsor a képernyő tetején, míg Windows esetén a képernyő alján jelenik meg.

Az állapotsorban csak korlátozott számú (a legtöbb képernyőn 38) karakter jeleníthető meg. A változó lehetővé teszi az AutoCAD által ismert adatok többségének megjelenítését. Ha az állapotsor tartalmát saját maga kívánja pontosan meghatározni, hasznosnak bizonyulhatnak a MODEMACRO matematikai, döntési és szerkesztési szolgáltatások.

A MODEMACRO változó

A felhasználó által definiált állapotsort a MODEMACRO rendszerváltozó vezérli. Induláskor az AutoCAD üres karakterláncot helyez el ebben a változóban. Ez a változó nem kerül elmentésre sem a rajzfájlban, sem a konfigurációs fájlban, sem másutt. Amennyiben azt szeretné, hogy a MODEMACRO változó a rajzok megnyitásakor mindig egy meghatározott értéket vegyen fel, akkor a MODEMACRO kívánt beállítását az *acad.lsp* fájlban definiált **S::STARTUP** függvény segítségével töltheti be.

A MODEMACRO egy a felhasználó számára biztosított karakterlánc típusú változó. Értéke bármilyen karakterlánc lehet, hosszát az AutoLISP és ADS változóira érvényes előírások, valamint az AutoLISP/AutoCAD kommunikációs puffertár mérete korlátozzák. A 460 karakter hosszúságú kifejezések általában minden rendszer esetében elfogadhatóak. A MODEMACRO beállítása a VÁLTOZÓK parancs segítségével lehetséges, amely tovább rövidíthető, és elég csak a MODEMACRO nevet beírni a Parancs promptra. A VÁLTOZÓK paranccsal különböző értékeket adhat a MODEMACRO változónak, s így kedvére kísérletezhet az állapotsor különböző formátumaival; az ily módon beírható karakterek száma azonban legfeljebb 255 lehet.

Amennyiben a MODEMACRO értéke az üres karakterlánc, az AutoCAD a szabvány állapotsort jeleníti meg. Ez az érték a parancssorból egy pont (.) bebillentyűzésével, illetve az AutoLISP **setvar** függvénye által átadott üres karakterlánccal ("") állítható be.

MODEMACRO definíciók

A MODEMACRO — amennyiben értéke nem üres karakterlánc — meghatározza, hogy mi jelenjen meg a állapotsorban. Jegyezze meg, hogy a képernyő felső sorának jobb oldalán megjelenő koordinátaértékek kijelzésére a MODEMACRO beállításának nincs hatása. Az állapotsor hosszát, azaz a megjeleníthető karakterek

maximális számát a megjelenítő eszköz meghajtója szabja meg. Ha a MODEMACRO hosszabb állapotsort próbál megjeleníteni, mint amekkora a képernyőn elfér, akkor a túlnyúló karakterek lemaradnak.

A MODEMACRO a legegyszerűbb (de a legkevésbé hasznos) esetben konstans szöveget tartalmaz. Ha Ön például a cége nevét kívánja az állapotsorba kiírni, egyszerűen be kell billentyűznie a következőket:

Parancs: **modemacro**
MODEMACRO új értéke, vagy . ha nincs <"">: **Frédi Bank és Grill**

Ez a MODEMACRO állandóan ugyanazt a szöveget írja ki, ily módon az állapotsor nem tükrözi az AutoCAD belső állapotváltozásait. Addig nem is történik változás, amíg a MODEMACRO értékét meg nem változtatja.

Ahhoz, hogy az állapotsor tükrözze az AutoCAD aktuális állapotát, DIESEL nyelven írt "makrókifejezéseket" kell elhelyeznie az állapotsorban. Ezek a makrókifejezések a következő formában írhatók:

```
$(valamifv, arg1, arg2, ...)
```

ahol a **valamifv** egy DIESEL függvény neve (hasonlóan az AutoLISP nyelvben használt függvénynevekhez), az *arg1*, *arg2* stb. pedig a függvény argumentumai, amelyeknek értelmezése a függvény definíciója szerint változik. Az AutoLISP nyelvtől eltérően a DIESEL makrókifejezések csak egyféle adattípust, a karakterláncot ismernek; A számokkal dolgozó makrók is karakteres formában kezelik, és szükség szerint oda-vissza konvertálják a számokat.

Definiáljunk egy érdekesebb állapotsort, például azt, amely az aktuális rajz nevét és fóliáját mutatja. A MODEMACRO felhasználásával ez a következőképpen írható:

Parancs: **modemacro**
MODEMACRO új értéke, vagy . ha nincs <"">: **$(getvar, dwgname)**
Fólia: $(getvar, clayer)

Megjegyzés A fenti, valamint a következő példákban a makrókifejezés folytatólagos sorokban látható, Önnek azonban ezt a promptterületre bebillentyűznie egyetlen hosszú karakterlánc formájában kell.

A **$(getvar,*váltnév*)** függvény segítségével bármely rendszerváltozót lekérdezheti, és annak aktuális értéke a makrókifejezés helyére kerül az állapotsorban. Ekkor, amennyiben fóliát vált, a MODEMACRO újra kiértékelésre kerül, és ha az értékében változás történt, akkor automatikusan az új fólianév jelenik meg a képernyő állapotsorában.

A kifejezések egymásba ágyazhatók, és tetszés szerinti bonyolultságúak lehetnek. Tételezzük fel például, hogy a Raszter aktuális értékét és fokokban megadott elforgatási szögét kívánja az

állapotsorban megjeleníteni. Az alábbi példában egymásba ágyazott kifejezéseket használunk az elforgatási szög radiánból fokra történő átszámítására, miközben leválasztjuk a szögérték törtrészét:

Parancs: **modemacro**
MODEMACRO új értéke, vagy . ha nincs <"">: Raszter: **$(getvar, snapunit) $(fix,$(*,$(getvar,snapang),$(/,180,3.14159)))**

Még jobb, ha ezeket az értékeket a MÉRTEGYS paranccsal beállított formátumban jelenítjük meg:

Parancs: **modemacro**
MODEMACROúj értéke, vagy . ha nincs <"">:
Raszter: $(rtos,$(index,0, $(getvar,snapunit))),$(rtos,$(index,1,$(getvar,snapunit))) $(angtos,$(getvar,snapang))

A DIESEL a bemenetére érkező kifejezéseket közvetlenül a kimenetére másolja mindaddig, amíg a "$" karakterrel vagy egy idézőjelbe tett karakterlánccal nem találkozik. A karakterláncokat idézőjelek közé téve megakadályozható azoknak a karaktersorozatoknak a kiértékelése, amelyek máskülönben DIESEL függvényként lennének értelmezhetők. Idézőjelet az idézőjeles karakterláncokon belül is elhelyezhet úgy, ha megkettőzi az idézőjelet. Ha például az aktuális fólia az ELRENDEZ, továbbá a MODEMACRO beállítása a következőképpen történt:

Parancs: **modemacro**
MODEMACROúj értéke, vagy . ha nincs <"">: **"$(getvar,clayer)= """"$(getvar,clayer)""""**

akkor az állapotsorban a következő jelenik meg:

```
$(getvar,clayer)="ELRENDEZ"
```

MODEMACRO definíciók az AutoLISP használatával

A MODEMACRO definíciók beállításához igénybe veheti az AutoLISP nyelvet. Az itt bemutatott példák elmenthetők ASCII formátumú szövegfájlokba, és betölthetők az AutoLISP **load** parancsával.

Az alábbi AutoLISP parancs olyan MODEMACRO beállítást definiál, amely az alapértelmezéssel azonos beállításokat tartalmaz. Minthogy az AutoLISP esetében a karakterláncok nem folytathatók sorról sorra, ezért az AutoLISP **strcat** függvénye használható arra, hogy a rövidebb részkifejezésekből összeállítsuk a teljes MODEMACRO karakterláncot.

```
(defun  C:ACADMODE ( )
  (setvar "modemacro"
     (strcat
```

```
        "Fólia $(substr,$(getvar,clayer),1,8)"
        "$(if,$(getvar,orthomode), Orto)"
        "$(if,$(getvar,snapmode), Raszter)"
        "$(if,$(getvar,tabmode), Tábla)"
        "$(if,$(=,$(getvar,tilemode),0),"
          "$(if,$(=,$(getvar,cvport),1), P)"
        ")"
    )
  )
)
```

Amennyiben ezt az AutoLISP rutint elmenti egy *acadmode.lsp* nevű fájlba, akkor a függvény betöltése, illetve végrehajtása után ugyanolyan állapotsort állít elő, mint amilyen az alapértelmezés szerinti állapotsor (nem ez a leghasznosabb alkalmazása ennek a szolgáltatásnak, csupán egy példa arra). Mivel semmilyen különbséget nem fog látni az állapotsor megjelenésében, ezért írjon "Fólia" helyett "F"-et; így meggyőződhet arról, hogy valóban a saját MODEMACRO beállítást használja.

A következőkben egy minta *acad.lsp* fájlt mutatunk be, amely az **S::STARTUP** függvényt használja fel arra, hogy a MODEMACRO változóban elhelyezze a *mode1.lsp* AutoLISP fájlban definiált karakterláncot .

```
;;; Minta ACAD.LSP fájl, amely az  S::STARTUP függvény
felhasználásával
;;; betölti a MODE1.LSP fájlt, amely egy MODEMACRO
karakterláncot definiál
(defun S::STARTUP ( )
  (load "mode1")
  (princ)
)
;;; Itt további AutoLISP fájlok betöltése következhet
```

Az alábbi AutoLISP fájl (*mode1.lsp*) betöltéskor a MODEMACRO rendszerváltozó felhasználásával olyan állapotsort definiál, amelyben a következők láthatók az F: után a Fólia nevének első nyolc karaktere; azután a rajz neve az elérési útvonal egy részével együtt; végül az aktuálisan bekapcsolt üzemmódok kezdőbetűje. A rajz nevének helye állandó, függetlenül a Fólia nevének hosszúságától.

A *mode1.lsp* minta AutoLISP fájl:

```
;;; MODE1.LSP
;;;
(setvar "modemacro"
   (strcat
    "L:$(substr,$(getvar,clayer),1,8)"
    "$(substr,        ,1,$(-,8,$(strlen,$(getvar,clayer)))) "
;;              ^^^^^^^^ Figyelje meg ezt a 8 szóközt
    "<.."
      "$(if,$(eq,$(getvar,dwgname),NÉVTELEN),NÉVTELEN,"
        "$(substr,$(getvar,dwgname),"
          "$(if,$(>,$(strlen,$(getvar,dwgprefix)),7),"
            "$(-,$(strlen,$(getvar,dwgprefix)),7),1"
          "),"
          "$(strlen,$(getvar,dwgname))"
        ")"
      ")"
    ">"
    "$(if,$(getvar,orthomode), O, )"
    "$(if,$(getvar,snapmode), S, )"
    "$(if,$(getvar,tabmode), T, )"
    "$(if,$(and,"
      "$(=,$(getvar,tilemode),0),$(=,$(getvar,cvport),1)), P)"
  )
)
```

Mint látható, a megfelelően alkalmazott tabulálás javítja az AutoLISP fájlok és a DIESEL kifejezések áttekinthetőségét.

DIESEL kifejezések a menükben

A DIESEL kifejezések beépíthetők a menüfájlokba, és ezzel további módszert kínálnak a makrók készítésére. Ezek a kifejezések a szabvány AutoCAD parancsoknak, AutoLISP és ADS rutinoknak, illetve más menümakróknak karakterlánc típusú értéket képesek válaszként megadni. Ez a visszaadott érték magában a menüben is felhasználható egy menücímke megjelenésének vagy tartalmának a megváltoztatására.

Amennyiben nem jártas a menük átalakításában, akkor a most következők előtt érdemes elolvasnia a 4. fejezetben a "Saját menük" című részt.

A menüpontban használt DIESEL kifejezésnek (további tudnivalók az "Almenük" című részben a 91. oldalon.) `"$szakasz=almenü"` formátumúnak kell lennie, ahol a szakasz neve `M`, az almenü pedig a kívánt DIESEL kifejezés. A menümakrót gyakorta egyszerűbb az AutoLISP segítségével elkészíteni.

A következő két példa ugyanarra a célra szolgáló menüpontokat mutat be; az egyik DIESEL-t, a másik AutoLISP-et használ:

DIESEL kifejezés a menüpontban:

```
[Ptér/Mtér]^C^C^P$M=$(if,$(=,$(getvar,cvport),1),mtér,ptér)
```

Ez a menüpont a következő AutoLISP kifejezést használja:

```
[-Ptér/Mtér-]^C^C^P (if (= (getvar "cvport") 1)(command "mtér")+
(command "ptér"))(princ)
```

Mindkét menüpont a Papírtér és a Modelltér közötti átkapcsolást teszi lehetővé (feltételezve, hogy a TILEMODE értéke 0), a DIESEL kifejezés azonban rövidebb, transzparens módon kiértékelhető, és az AutoLISP **princ** függvényének a használatát sem igényli. Ha mindkét példából elhagyjuk a speciális `^P` karaktert (amely a MENUECHO be- és kikapcsolására szolgál), akkor a DIESEL kifejezés csak a ténylegesen kiadott parancsot jeleníti meg, míg az AutoLISP kifejezés az egész programsort.

Minthogy a DIESEL kifejezés visszatérési értéke egy szöveges karakterlánc, átadható válaszul az AutoLISP valamely **getxxx** függvényhívására. Ezáltal, a menüpontok képesek arra, hogy kiértékeljék a rajzolás aktuális körülményeit, és átadják az értéket egy AutoLISP rutinnak.

A következő példa négy feltételezésen alapul:

- Az AutoLISP rutin betöltése előzőleg megtörtént.
- A menürészlet része az aktuális menünek.
- A beillesztendő szimbólumok egy egység magasak és egy egység szélesek.
- A DIMSCALE változó értéke a rajz méretarányának megfelelő (azaz ha a rajz 1"=10' léptékben lesz kirajzolva, akkor a méretarány-tényező 120, egy 1/4"=1' léptékű rajz esetében pedig 48).

Amennyiben betölti és végrehajtja a mintaként bemutatott alábbi AutoLISP rutint, az AutoCAD a megfelelő méretben és a kívánt helyre illeszti be a szimbólumot. Kirajzoláskor a szimbólumok a kívánt méretűek lesznek (feltéve, hogy a rajzot a DIMSCALE által meghatározott méretarányban rajzolja ki).

Minta AutoLISP rutin:

```
(defun C:SZIMBEILL ( )
  (setq sym
    (getstring "\nAdja meg a szimbólum nevét: "))    ; A szimbólum nevét kéri
  (menucmd "s=szimméret")                          ; a szimméret almenüre váltja a
                                                     képernyőmenüt
  (setq mér
    (getreal "\nVálassza ki a szimbólum méretét: ")     ;aszmbólummméretétkéri
    p1 (getpoint "\nBeillesztés pontja: "))   ; A A beillesztés pontját kéri
  (command "beill" szim p1 mér mér 0)          ; Kiadja a BEILL parancsot
                                                ; megadott szimbólum beillesztési
                                                  pont és méret felhasználásával
  (menucmd "s=")                                ; Az előző képernyőmenüre kapcsol
                                                  vissza
  (princ)                                       ; Csendben kilép
```

Megjegyzés A rendszeresen használt rutinokba olyan hibaellenőrzést is érdemes beépíteni, amely megvizsgálja a felhasználó által megadott adat érvényességét.

Az iménti menüfájlban szereplő DIESEL kifejezések megszorozzák a DIMSCALE aktuális értékét a kívánt értékkel, és egy ennek megfelelő méretarány-tényezőt adnak eredményül. Ez egy hasonló AutoLISP kifejezéssel nem tehető meg, ugyanis az AutoLISP kifejezés visszatérési értéke *nem* adható át válaszul egy **getxxx** függvényhívásnak (például az előbbi minta AutoLISP rutinban a **getreal** függvénynek).

Következzék itt egy menüfájl kivonata:

```
**szimméret 3
[ MÉRETEK ]

[ 3/8" ]$M= $(*, $(getvar,dimscale),0.375)
[ 1/2" ]$M=$(*,$(getvar,dimscale),0.5)
[ 5/8" ]$M=$(*,$(getvar,dimscale),0.625)
```

A DIESEL kifejezések által szolgáltatott karakterláncok a legördülő menük tételcímkéinek is átadhatók, lehetőséget biztosítva azok elhalványítására, illetve megjelenítésük egyéb módon történő megváltoztatására. Ha a legördülő menü címkéjében DIESEL kifejezést kíván használni, akkor az első karakternek a "$" jelnek kell lennie.

Amennyiben az aktuális Fólia a BÁZIS, és a menüfájl valamely ***POP*n szakaszában a

```
[$ (eval,"Aktuális fólia: " $(getvar,clayer))]
```

címke szerepel, akkor az adott legördülő menüben az

Aktuális fólia: BÁZIS

menüpont jelenik meg, amely mindig követi az aktuális fólia változását. Ez a módszer arra is felhasználható, hogy interaktív módon megváltoztassa a legördülő menüben megjelenő szöveget egy

AutoLISP rutin felhasználásával. A rutin elhelyezi a kívánt szöveget a USERS1-5 rendszerváltozók valamelyikében, ami a menüpont címkéjében elhelyezett DIESEL makró segítségével kiolvasható.

Megjegyzés A legördülő és kurzormenük szélességét az AutoCAD a menüfájl betöltésekor meghatározza, ezért a DIESEL kifejezésekkel a menü betöltése után előállított vagy módosított tételcímkéknek a már adott szélességi méreten túlnyúló része nem fog látszani.

Amennyiben előre tudja, hogy egy DIESEL kifejezéssel előállított címke túl széles lesz ahhoz, hogy elférjen az aktuális menün belül, akkor az alábbi példa felhasználásával biztosíthatja, hogy a menü szélessége alkalmazkodjon a címke kívánt méretéhez. Ez a példa a USERS3 rendszerváltozó aktuális értékének első tíz karakterét jeleníti meg:

```
[$ (eval,"Aktuális érték: " $(getvar,users3))+
 $(if, $(eq,$(getvar,users3),""), 10 szóköz )]^C^Cusers3
```

A menü címkéjének végéhez nem illeszthet szóközöket, mivel azokat az AutoCAD a menü betöltésekor nem veszi figyelembe. A címke szélességének növelésére szolgáló szóközök csakis egy DIESEL kifejezésen belül helyezhetők el.

Az alábbi példa ugyanazt a DIESEL kifejezést alkalmazza címkeként, illetve a menüpont részeként is. Ez praktikus módszer az aktuális nap nevének és a dátumnak a rajzba történő beillesztésére.

```
[$(edtime,$(getvar,date),DDD", "D MON YYYY)]^C^Cszöveg +
\\\ $M=$(edtime,$(getvar,date),DDD", "D MON YYYY);
```

A legördülő menü címkéinek elhalványítására vagy megjelölésére is használhat DIESEL makrót. A következő legördülő menüben a RADÍR tétel halvány (és nem választható ki) egy másik parancs működése közben. Amikor nincs folyamatban lévő parancs, akkor a szöveg normál módon jelenik meg.

```
[ $(if,$(getvar,cmdactive),~)RADIR]radír
```

Hasonló elv szerint lehet egy jelet elhelyezni a legördülő menü egyik tétele mellett, vagy interaktív módon kicserélni a jelölésre használt karaktert.

Megjegyzés A DIESEL kifejezések használata a menük címkéiben platformfüggő. Egyes platformok esetében a menüfájl betöltése után már nem kerül sor a menük címkéinek aktualizálására.

DIESEL kifejezések az AutoLISP rutinokban

Az AutoLISP **menucmd** függvényének hívásával AutoLISP rutinokban is használhatja a DIESEL kifejezéseket. Ennek formája hasonló ahhoz, mint amikor — az előbb ismertetett módon — menüpontokban használja a DIESEL kifejezéseket.

A következő programrészlet az `a_idő` változót az aktuális időponttal teszi egyenlővé:

```
(setq c_time (menucmd "M=$(edtime,$(getvar,date),HH:MM a/p)"))
```

Az AutoLISP-et használhatja A DIESEL kifejezések vizsgálatára (tesztelésére). Az alábbi mintarutin egy új parancsot definiál, amely lehetővé teszi, hogy DIESEL kifejezéseket írjon a parancssorba:

```
;;; DIESEL.LSP
;;; Lehetővé teszi a DIESEL kifejezések megadását a
;;;parancssorból

(defun C:DIESEL ( / dsl )
  (while (/= dsl "M=")
    (setq dsl (strcat "M=" (getstring T "\nDIESEL: ")))
     (princ (menucmd dsl))
  )
  (princ)
)
```

Ha ennek a rutinnak a definiálását követően a parancssorba beírja a **diesel** parancsot, akkor a DIESEL prompt jelenik meg. Erre a promptra bármilyen DIESEL kifejezést bebillentyűzhet; ha a kifejezés érvényes, akkor az eredményt kapja vissza, ha érvénytelen, akkor egy megfelelő DIESEL hibaüzenet jelenik meg. A fenti rutin ismételten megjeleníti a DIESEL promptot mindaddig, amíg Ön üres választ nem ad (például a ↵ leütésével).

Hibakeresés a DIESEL kifejezésekben—MACROTRACE

A MACROTRACE rendszerváltozó a DIESEL kifejezések hibáinak kereséséhez nyújt segítséget. Alapértelmezés szerinti értéke 0 (Ki). Amennyiben értékét 1-re (Be) állítja, akkor *minden* DIESEL kifejezés kiértékelésre kerül, beleértve a menükben és az állapotsorban használt kifejezéseket is.

Mivel a MACROTRACE valamennyi DIESEL kifejezés kiértékelését nyomon követi, számítania kell arra, hogy nem csak annak a kifejezésnek az eredményét kapja vissza, amelyet ellenőrizni kívánt. Sok felesleges DIESEL kiértékelés elkerülhető, ha a MODEMACRO változónak üres karakterláncot ad értékül.

DIESEL karakterlánc-függvények katalógusa

A DIESEL függvények alkalmasak státusz-lekérdezésre, számítási műveletek végrehajtására és megjelenítésre. Az alábbiakban felsoroljuk a rendelkezésre álló függvényeket.

Megjegyzés Valamennyi függvényre érvényes, hogy a paramétereik száma nem lehet több tíznél, beleszámítva a függvény nevét is. E határérték túllépésére egy DIESEL hibaüzenet a válasz.

+ (összeadás)

Az *ért1*, *ért2*, ..., *ért9* számok összegével tér vissza.

$(+, ért1 [, ért2, ..., ért9])

Ha a vastagság aktuális értéke 5, akkor a következő DIESEL karakterlánc értéke 15.

```
$(+, $(getvar,vastagság),10)
```

– (kivonás)

Az *ért9*-től *ért2*-ig terjedő számokat *ért1*-ből kivonja, s az eredménnyel tér vissza.

$(-, ért1 [, ért2 , ..., ért9])

*(szorzás)

Az *val1*, *val2*, ..., *val9* számok szorzatával tér vissza.

$(*, ért1 [, ért2, ..., ért9])

/ (osztás)

Az *ért1* számot elosztja az *ért2*, ..., *ért9* számokkal, és az eredménnyel tér vissza.

$(/, ért1 [, ért2, ..., ért9])

= (egyenlő)

Ha az *ért1* és az *ért2* számok egyenlőek, akkor 1, különben 0 értékkel tér vissza.

$(=, ért1, ért2)

< (kisebb, mint)

Ha az *ért1* szám kisebb, mint *ért2*, akkor 1, egyébként 0 értékkel tér vissza.

$(< , ért1, ért2)

Az alábbi kifejezés kiolvassa a HPANG aktuális értékét; ha ez az érték kisebb, mint a USERR1 rendszerváltozóban tárolt érték, akkor az eredmény 1 lesz. Feltételezve, hogy a USERR1 változóban tárolt érték 10.0, és a HPANG aktuális értéke 15.5, a következő karakterlánc 0 értékkel tér vissza.

```
$(<, $(getvar,hpang),$(getvar,userr1))
```

> (nagyobb, mint)

Ha az *ért1* szám nagyobb, mint *ért2*, akkor 1, egyébként 0 értékkel tér vissza.

$(>, ért1, ért2)

!=(nem egyenlő)

Ha az *ért1* és az *ért2* számok nem egyenlőek, akkor 1, egyébként 0 értékkel tér vissza.

$(!=, ért1, ért2)

<= (kisebb egyenlő)

Ha az *ért1* szám kisebb vagy egyenlő *ért2-vel* akkor 1, egyébként 0 értékkel tér vissza.

$(<=, ért1, ért2)

>= (nagyobb egyenlő)

Ha az *ért1* szám nagyobb vagy egyenlő *ért2-vel* akkor 1, egyébként 0 értékkel tér vissza.

$(>=, ért1, ért2)

és

Az *ért1*-től *ért9*-ig terjedő egész számokon bitenként elvégzett logikai AND művelet eredményével tér vissza.

$(and, ért1 [, ért2,..., ért9])

angtos

A megadott formátumú és pontosságú szögértékkel tér vissza.

$(angtos, érték [, mód, pontosság])

A megadott értéket az analóg AutoLISP függvénnyel azonosan definiált *mód* és *pontosság* paraméterek által meghatározott szögformátumba írja át (a *mód* értékei az alábbi táblázatban láthatók). A mód és a *pontosság* elhagyása esetén a függvény a MÉRTEGYS paranccsal kiválasztott aktuális értékeket veszi alapul

Szögmértékegységek

Mód értéke	*Karakterlánc formátuma*
0	Fok
1	Fok/perc/másodperc
2	Grad
3	Radián
4	Geodéziai mértékegységek

edtime

Egy megadott alak szerint formátumozott dátummal és idővel tér vissza.

$(edtime, idő, alak)

Az *idő* által megadott Juliánusz-dátumot (kiolvasható például a $(**getvar,date**) kifejezéssel) a megadott *alak* szerinti formátumba írja át. Az *alak* argumentumot formátum-kifejezések alkotják, amelyek helyére a függvény a dátumot és az időpontot meghatározott alakban helyettesíti be. A formátum-kifejezések által nem értelmezhető karakterek változatlanul jelennek meg a $(**edtime**) eredményében. A formátumkifejezések definíciója az alábbi táblázatban látható. Az alapul vett dátum és időpont 1993. szeptember 2., csütörtök, 4:53:17.506.

Az edtime függvény formátum-kifejezései

Formátum	*Eredmény*	*Formátum*	*Eredmény*
D	2	H	4
DD	02	HH	04

Az edtime függvény formátum-kifejezései (*folytatás*)

Formátum	*Eredmény*	*Formátum*	*Eredmény*
DDD	Csüt	MM	53
DDDD	Csütörtök	SS	17
M	9	MSEC	506
MO	09	AM/PM	AM
MON	Sze	am/pm	am
MONTH	Szeptember	A/P	A
YY	93	a/p	a
YYYY	1993		

Megjegyzés A teljes "AM/PM" kifejezést szükséges megadni, amint azt a táblázat mutatja. Abban az esetben, ha csak az "AM" szerepel, akkor az "A" betűt a függvény változatlanul visszaadja, az "M" pedig az aktuális hónapot fogja eredményezni.

Amennyiben szerepel az alakban az "AM/PM" kifejezések valamelyike, akkor a "H" és "HH" kifejezések az órákat 12 órás periódusban (12:00-12:59-1:00-11:59) fogják megjeleníteni 24 órás periódus (00:00-23:59) helyett.

Az alábbi példában a táblázat esetében is használt dátumot és időpontot vettük alapul (figyelje meg, hogy a vesszőt idézőjelek közé kell tenni, nehogy a függvény argumentum-elválasztójelnek tekintse):

```
$(edtime, $(getvar,date),DDD"," DD MON YYYY - H:MMam/pm)
```

eredménye:

1993.Sze.02., Csü.-4:53am

Amennyiben az *idő* argumentum értéke 0, akkor a függvény a legkülső makró végrehajtásának pillanatát tekinti dátumnak, illetve időpontnak. Ezzel elkerülhetők a többszöri, hosszadalmas és időigényes $(getvar,date) hívások, valamint biztosítható, hogy a több $(edtime) makróval összeállított karakterláncok mind ugyanazt az időpontot vegyék alapul.

eq

Ha az *ért1* és *ért2* karakterláncok azonosak, akkor 1, egyébként 0 értékkel tér vissza.

$(eq, *ért1*, *ért2*)

Az alábbi kifejezés kiolvassa az aktuális Fólia nevét; amennyiben a név egyezik a USERS1 rendszerváltozóban tárolt karakterlánccal, akkor az eredmény 1 lesz. Feltételezve, hogy a USERS1 változóban tárolt karakterlánc az "ALKATR12", és az aktuális fólia ezzel megegyezik:

```
$(eq, $(getvar,users1),$(getvar,clayer))
```
eredménye 1

eval

A *karlnc* karakterláncot átadja a DIESEL kiértékelőnek, és a kiértékelés eredményével tér vissza.

$(eval, karlnc)

fix

Az *érték* valós számot a törtrészének elhagyásával egész típusú számra csonkol.

$(fix, érték)

getenv

A *váltnév* környezeti változó értékével tér vissza.

$(getenv, váltnév)

Ha a megadott nevű változó nem létezik, akkor az üres karakterlánccal tér vissza.

getvar

A *váltnév* által megadott rendszerváltozó értékével tér vissza.

$(getvar, váltnév)

if

Feltételesen értékeli ki a kifejezéseket.

$(if, kif, *ha_igaz [, ha_hamis]*)

Ha a *kif* nem nulla, akkor kiértékeli a *ha_igaz* kifejezést, és annak eredményével tér vissza. Egyébként a *ha_hamis* kifejezést értékeli ki, és annak eredményével tér vissza. Fontos megjegyezni, hogy csak a *kif* által kiválasztott argumentumot értékeli ki.

index

Visszaadja a vesszővel elválasztott karakterlánc tagjait.

$(index, melyik, *karakterlánc*)

Feltételezi, hogy a *karakterlánc* argumentum egy vagy több értéket tartalmaz, melyeket vessző (a makró argumentum-elválasztójel) határol el egymástól. A *melyik* argumentum választja ki ezen értékek közül azt az egyet, amellyel a függvény visszatér. Az értékek indexelése nullával kezdődik. Ezt a függvényt leggyakrabban arra használjuk, hogy egy, a `$(getvar)` által kiolvasott pont koordinátái közül az *X*, *Y* vagy *Z* koordinátát megkapjuk.

Az alkalmazások a USERS1-5 rendszerváltozókban tárolt, vesszővel tagolt karakterláncok lekérdezésére használhatják ezt a függvényt.

linelen

A megjelenítő eszköz által a lehető leghosszabb állapotsor hosszával, azaz az abban elhelyezhető karakterek maximális számával tér vissza

`$(linelen)`

Ezt az állapotsor formátumának — a megjelenítő eszköz adottságaitól függő — beállítására használhatja, de csak akkor látja hasznát, ha az állapotsort a MODEMACRO segítségével konfigurálja.

nth

A *melyik* által kiválasztott argumentum kiértékelésének az eredményével tér vissza.

`$(nth, melyik, arg0 [, arg1,…, arg7])`

Ha *melyik* értéke 0, akkor az nth *ért0* értékével tér vissza, és így tovább. Figyelje meg a **`$(nth)`** és a **`$(index)`** közötti különbséget. A **`$(nth)`** a függvény argumentumainak sorából az egyiknek az értékét adja vissza; ezzel szemben a $(index) az egyetlen argumentumként átadott karakterláncban levő, vesszővel elválasztott értékek közül emel ki egyet. Csak a *melyik* által kiválasztott argumentumot értékeli ki.

or

Az *ért1*-től *ért9*-ig terjedő egész számokon bitenként elvégzett logikai OR művelet eredményével tér vissza.

`$(or, ért1 [, ért2,…, ért9])`

rtos

A megadott formátumú és pontosságú valós értékkel tér vissza.

`$(rtos, érték [, mód, pontosság])`

A megadott értéket az analóg AutoLISP függvénnyel azonosan definiált *mód* és *pontosság* paraméterek által meghatározott valós számformátumba írja át. A *mód* és a *pontosság* elhagyása esetén a

függvény a MÉRTEGYS paranccsal kiválasztott aktuális értékeket veszi alapul.

strlen

A *karakterlánc* hosszával, azaz a benne szereplő karakterek számával tér vissza.

```
$(strlen, karakterlánc)
```

substr

A *karakterláncnak* azzal a részkarakterláncával tér vissza, amely a `start` karakterpozíciónál kezdődik, és *hossz* számú karaktert tartalmaz.

```
$(substr, karakterlánc, kezdő [, hossz])
```

A karakterláncban a karakterek számozása 1-gyel kezdődik. A *hossz* argumentum elhagyása esetén a karakterlánc teljes fennmaradó hosszát visszaadja.

upper

A *karakterlánc* karaktereit nagybetűkké alakítja át az aktuális platformon érvényes szabályok szerint.

```
$(upper, karakterlánc)
```

xor

Az *ért1*-től *ért9*-ig terjedő egész számokon bitenként elvégzett logikai XOR művelet eredményével tér vissza.

```
$(xor, ért1 [, ért2,…, ért9])
```

Hibaüzenetek

Általában igaz, hogy ha hibát vét egy DIESEL kifejezésben, egyértelműen látható, hol történt baj. A hiba természetétől függően a DIESEL a kimenő adatfolyamban hibajelzést helyez el.

DIESEL hibaüzenetek

Hibaüzenet	*Leírás*
$?	Szintaktikai hiba (általában a jobboldali zárójel vagy a karakterlánc lezárásának a hiánya)

DIESEL hibaüzenetek

Hibaüzenet	*Leírás*
$*(függv,??)*	Helytelen argumentumok a `függv` függvényhívásban
$*(függv)??*	A `függv` függvény ismeretlen
$(++)	Az eredményül kapott karakterlánc túl hosszú—a kiértékelés csonkolva

6

PROGRAMOZÁSI FELÜLETEK

A parancssor és a menük kínálta lehetőségeken túl az AutoCAD parancsírási lehetőséget és a felhasználó által definiált programozási felületeket is kínál, amelyek rajzok és adatbázisok kezelésére használhatók. Ez a fejezet a felhasználó által definiálható AutoLISP és ADS programozói felületeket mutatja be. A használt felület típusa az alkalmazás támasztotta követelményektől és az Ön programozási gyakorlatától függ. Az AutoCAD a szabványos parancskészlet kiegészítésére az ARX alkalmazási felületet is igénybe veszi.

Ez a fejezet bemutatja a különböző programozási felületek fogalmát és használatát, valamint leírja, hogyan tölthetünk be és hogyan használhatunk más fejlesztők által írt alkalmazásokat.

Az AutoLISP programozás részleteit jelen könyv II., "AutoLISP referencia" című részében találja meg. Az ADS programozási nyelv leírása az AutoCAD Fejlesztõi rendszerprogramozói kézikönyv című kötetben olvasható. Az AutoLISP és az ADS programok is használhatnak párbeszédpaneleket az alkalmazásokban. A programozható párbeszédpanelek leírását ezen útmutató III. részében, a "Programozható párbeszédpanel referencia" című fejezetben találja.

Parancs-forgatókönyvek

Az AutoCAD *forgatókönyv*-szolgáltatása lehetővé teszi a parancsok szövegfájlból történő beolvasását. Ezzel a szolgáltatással előre meghatározott parancssorozatokat hajthat végre; ezeket meghívhatja az AutoCAD indításával egyidejűleg (az **acad** utasítás egy speciális formájával), vagy elindíthatja az AutoCAD programból az FKÖNYV paranccsal. A forgatókönyv-fájlok segítségével egyszerűen és könnyen megjeleníthet folyamatosan futó képeket termékbemutatók vagy kiállítások alkalmával.

Forgatókönyv-fájlokat az AutoCAD programtól függetlenül, egy editorral (például DOS Edit vagy Windows Notepad) vagy egy szövegszerkesztővel (például Word) hozhat létre. A fájl kiterjesztése minden esetben *.scr*.

A forgatókönyv-fájlokban megjegyzés formájában a fájl tartalmának magyarázata vagy egyéb lényeges információ is elhelyezhető. A pontosvesszővel (;) kezdődő sorokat az AutoCAD megjegyzésnek tekinti, és a forgatókönyv-fájl feldolgozásakor nem veszi figyelembe.

Amikor az AutoCAD egy forgatókönyvből kapja a parancsot, a PICKADD és PICKAUTO rendszerváltozók feltételezett értéke 1, illetve 0. Ez biztosítja a kompatibilitást az AutoCAD előző verzióival, és megkönnyíti az alkalmazás készítőjének dolgát (mivel nem kell ellenőriznie ezeknek a változóknak a beállítását).

Az AutoCAD VISSZA parancsa a forgatókönyvet egyetlen *csoportnak* tekinti, így hatása egyetlen V paranccsal érvényteleníthető. Ugyanakkor a forgatókönyv minden parancsa előidéz egy bejegyzést az AutoCAD Vissza "naplójában", ami kissé lelassíthatja a forgatókönyv feldolgozását. A parancs Vezérlés Semmi opciójával azonban tetszés szerint ki is kapcsolhatja a Vissza lehetőséget még a forgatókönyv elindítása előtt (vagy magában a forgatókönyv-fájlban). A forgatókönyv lefutása után ezt ne felejtse el visszaállítani a normális állapotba (VISSZA Vezérlés Mind).

Újabb FKÖNYV parancs meghívásakor az aktuális forgatókönyv lezárul.

A forgatókönyv elindítása az AutoCAD betöltésekor

Amennyiben a forgatókönyvet az AutoCAD első betöltésekor kívánja elindítani, ezt a következő formátumú utasítással teheti meg:

lemezmeghajtó> **acad rajznév** ***forgatókönyv-fájl***

Az **acad** programot meghívó sorban a forgatókönyv nevének a második fájlnévnek kell lennie; a program az *.scr* kiterjesztés

feltételezi. Ha az AutoCAD nem találja a forgatókönyv-fájlt, akkor jelzi, hogy a fájl megnyitása nem lehetséges.

Tételezzük fel, hogy valahányszor egy új rajz szerkesztésébe kezd, első lépésben azonnal bekapcsolja a ponthálót, a vonaltípusok globális méretarányát 3.0-ra állítja, és aktuális fóliaként a 0 fóliát választja ki, vörös színnel. Mindezt elvégezheti egy prototípus-rajz segítségével, de a jelen fejezet témája miatt ezt most inkább a *setup.scr* fájlban tárolt alábbi forgatókönyvvel hajtsa végre.

```
háló be                          Pontháló bekapcsolása
vtlépték 3.0                     Vonaltípusok méretezésének beállítása
fólia beá 0 szín vörös 0         Az aktuális fólia és színének beállítása
                                 Üres sor a FÓLIA parancs lezárásához
```

Ezt követően egy *xkrax* nevű új rajz létrehozásához az AutoCAD programot a következőképpen kell elindítani:

lemezmeghajtó> **acad xkrax setup**

Ez az alapértelmezés szerinti fájlnevet *xkrax*-ra állítja, és a parancsokat a *setup.scr* fájlból kezdi kiolvasni. Ez a forgatókönyv egy új rajzot hoz létre az alapértelmezés szerinti névvel (*xkrax*), majd végrehajtja az alapbeállításra vonatkozó parancsok sorozatát. Amikor elérte a forgatókönyv-fájl végét, az AutoCAD megjeleníti a Parancs promptot.

Természetesen ahhoz, hogy el tudja helyezni a forgatókönyvben a megfelelő válaszsorozatot, nagyon jól kell ismernie az AutoCAD promptsorozatait. Megjegyezzük, hogy a promptok (sőt maguk a parancsnevek is) megváltozhatnak az AutoCAD későbbi kiadásaiban, ezért az AutoCAD újabb verziójára történő áttéréskor megtörténhet, hogy a forgatókönyvén bizonyos változtatásokat kell elvégeznie. Hasonló okokból célszerű kerülni a rövidítéseket; a parancsok jövőbeni kiegészítése esetén ez félreértésekhez vezethet. Ne feledkezzen meg arról sem, hogy a forgatókönyvekben minden egyes szóköznek jelentősége van, hiszen az AutoCAD a parancs vagy az adatbevitel lezárásaként a szóköz vagy a ↵ billentyű lenyomását várja.

Egy forgatókönyv segítségével az automatikusan vezérelt nyomtatást (Freeplot) is használhat, vagy újrakonfigurálhatja az AutoCAD programot az operációs rendszerből. A következő formátum meghívja a Freeplot funkciót és meghatározza a forgatókönyv nevét:

lemezmeghajtó> **acad –p *forgatókönyv-fájl***

Az alábbi forgatókönyv a *mydwg.dwg* nevű rajzot rajzolja ki. A Freeplot eljárásban használt forgatókönyvben a rajz nevének közvetlenül a

PLOT parancsot kell követnie. Ez a fogatókönyv-példa feltételezi, hogy az AutoCAD jelenlegi konfigurációja szerint a fájlba rajzol.

```
PLOT                    Kiadja a PLOT parancsot
MYDWG                   Megadja a megnyitandó rajzot
DISPLAY                 Megadja a rajz kirajzolandó részét
                        Elfogadja a jelenlegi alapértelmezéseket
(üres sor)
MYDWG                   Meghatározza a plotfájl nevét
```

A kirajzoltatásra írt forgatókönyv fejlesztése és tesztelése alatt hasznos a CMDDIA rendszerváltozót 0-ra állítani. Ez lehetővé teszi minden prompt átlépését, ahogy azt a forgatókönyv teszi.

Az alábbi rendszerprompt-szintaxis az újrakonfiguráláshoz ad meg egy forgatókönyvet.

lemezmeghajtó> **acad –r** ***rajznév*** **forgatókönyv-fájl**

Diabemutatók készítése

A forgatókönyvek jól felhasználhatóak a diabemutatók készítésénél. A diaképek bemutatásának sebességét általában a diafájl beolvasásához szükséges lemezhozzáférési idő korlátozza. Lehetőség van azonban arra, hogy előre betöltse a memóriába a következő diát a lemezről, miközben a közönség még az aktuális diát szemléli, s így a következő lépésben a diát a memóriából gyorsan megjelenítheti.

Egy dia előzetes betöltéséhez a BEDIA parancsban a fájlnév elé írjon egy csillagot. A következő BEDIA parancs érzékelni fogja, hogy a dia már be van töltve, és megjeleníti anélkül, hogy rákérdezne a nevére. Példaként tekintse meg a következő forgatókönyvet:

```
BEDIA DIA1      diabemutató kezdete, az 1. dia betöltése
BEDIA *DIA2     2. dia előkészítése
SZÜNET 2000     a közönség megtekintheti az 1.diát
BEDIA           2. dia megjelenítése
BEDIA *DIA3     3. dia előkészítése
SZÜNET 2000     a közönség megtekintheti a 2.diát
BEDIA           3. dia megjelenítése
SZÜNET 3000     a közönség megtekintheti a 3.diát
ÚJRAFUT         Ciklus
```

A következő dia betöltéséhez szükséges lemezhozzáférési időt átfedi az aktuális dia megtekintési idejét. Ezen kívül további késleltetést is beiktattunk.

AutoLISP

Az AutoLISP a LISP programozási nyelvre épül, amely könnyen elsajátítható, ugyanakkor igen hatékony eszköz. Az AutoCAD beépített LISP értelmezővel rendelkezik, amely lehetővé teszi, hogy a

felhasználó az AutoLISP kifejezéseket közvetlenül a parancssorba beírja, vagy külső fájlból töltse be. Az AutoLISP alkalmazások (rutinok) sokféleképpen képesek együttműködni az AutoCAD rendszerrel. E rutinok kérhetik adatok bevitelét a felhasználótól, közvetlenül elérhetik a beépített AutoCAD parancsokat, valamint módosíthatnak vagy létrehozhatnak rajzelemeket a rajzok adatbázisában. AutoLISP rutinok készítésével új, szakágspecifikus parancsokkal egészítheti ki az AutoCAD rendszerét. Számos szabványos AutoCAD parancs is AutoLISP alkalmazás.

Mivel az AutoCAD közvetlenül olvassa az AutoLISP kódot, nincs szükség fordításra: az AutoLISP kódot a parancssorba beírva azonnal láthatja az eredményt. Ezáltal az AutoLISP nyelvvel könnyű kísérletezni, tekintet nélkül a programozó gyakorlatára. Ha már jártasságot szerzett az AutoLISP használatában, rájön, milyen jól alkalmazható az AutoCAD eredeti parancsainak kiegészítésére.

Amennyiben nem érdeklődik az AutoLISP programok írása iránt, akkor is sok hasznos rutint talál az AutoCAD csomagban (a *sample* és *support* alkönyvtárakban). Hozzájuthat még független fejlesztőktől származó, továbbá "shareware" termékként forgalmazott rutinokhoz is. Nagyban javíthat munkájának hatékonyságán, ha tudja, hogyan kell betölteni és használni ezeket a rutinokat.

Megjegyzés Amikor az AutoCAD az AutoLISP **command** függvényétől kapja a parancsot, a PICKADD és PICKAUTO rendszerváltozók feltételezett értéke 1, illetve 0. Ez biztosítja a kompatibilitást az AutoCAD előző verzióival, és megkönnyíti az alkalmazás készítőjének a dolgát (mivel nem kell ellenőriznie ezeknek a változóknak a beállítását).

AutoLISP alkalmazások használata

Az AutoLISP programokat *.lsp* kiterjesztésű ASCII szövegfájlok tartalmazzák. Ezek a fájlok általában egy fejrésszel kezdődnek, amely ismerteti a fájlban található rutint vagy rutinokat, azok használatát, és egyéb tudnivalókat. A fejléccel a szerző kilétét és a rutin felhasználásának jogi vonatkozásait is dokumentálhatja. A megjegyzéseket egy pontosvessző előzi meg. Ezek a fájlok bármely, ASCII szövegfájl létrehozására alkalmas editorral vagy szövegszerkesztővel megtekinthetők és szerkeszthetők.

Ahhoz, hogy egy AutoLISP alkalmazást használni lehessen, azt előbb be kell tölteni. Egy alkalmazás betöltésére használhatja az ALKTÖLT parancsot (amely maga is egy AutoLISP alkalmazás) vagy az AutoLISP **load** függvényét (további tudnivaló „ALKTÖLT" címnél az on-line súgóban). Egy AutoLISP alkalmazás betöltése az AutoLISP kódot a .lsp fájlból a rendszermemóriába tölti.

Ha egy alkalmazást a load függvénnyel tölt be, ez eleve azzal jár, hogy AutoLISP kifejezést ír be a parancssorba. Ha a betöltés sikerült, akkor a

load a fájl utolsó kifejezésének értékét adja vissza a parancssorban. Ez rendszerint a fájlban definiált utolsó függvény neve, esetleg rövid útmutatás az újonnan definiált függvény használatára vonatkozóan. Ha a **load** művelet sikertelen volt, akkor egy AutoLISP hibaüzenetet jelenik meg. Ez bekövetkezhet a fájlban szereplő helytelen kifejezés miatt, vagy azért, mert rossz fájlnevet írt a parancssorba. A **load** függvény szintaxisa a következő:

```
(load fájlnév [hibaeset])
```

Ebből kitűnik, hogy a **load** függvénynek két argumentuma van; egyik a *fájlnév*, amelyre feltétlenül szükség van, a másik a *hibaeset*, amely elhagyható (opcionális). Az AutoLISP fájl parancssorból történő betöltésekor általában elég csak a *fájlnév* argumentumot megadni. Az alábbi példa a *myfile.lsp* nevű AutoLISP fájl betöltését mutatja:

Parancs: **(load "myfile")**

Az *.lsp* kiterjesztést nem szükséges megadni. Ez a formátum minden olyan *.lsp* fájl betöltéséhez megfelelő, amely az AutoCAD aktuális könyvtárútvonalán található. (További tudnivaló a "A könyvtárkeresési útvonal" című résznél, a 18. oldalon található.)

Olyan AutoLISP fájl betöltéséhez, amely nem ezen könyvtárak valamelyikében található, *fájlnév* argumentumként a teljes útvonal/fájlnév meghatározást meg kell adnia. Az útvonal megadásakor normál törtjelet (/) vagy két visszahajló törtjelet (\\ ) használjon a könyvtárak tagolására, mivel a visszahajló törtjelnek speciális jelentése van az AutoLISP nyelvben.

Parancs: **(load "d:/files/morelisp/newfile")**

Automatikus betöltés és végrehajtás

Amennyiben egy *könyvtárat* állít össze a hasznos AutoLISP rutinokból, valószínűleg szeretné azokat automatikusan betölteni az AutoCAD minden indításakor. Felmerülhet olyan igény is, hogy bizonyos parancsok vagy függvények a rajzszerkesztés meghatározott pontján kerüljenek végrehajtásra.

Két olyan fájl van, amelyek tartalmát az AutoCAD automatikusan betölti: az *acad.lsp* és az Ön által aktuálisan használt menühöz társított *.mnl* fájl. Amennyiben ezen fájlok valamelyikében szerepel a speciális típusú S::STARTUP függvény definíciója, akkor ezt a rutint az AutoCAD automatikusan végrehajtja, amikor egy rajzhoz hozzákezd. Az S::STARTUP függvény megtalálható az "S::STARTUP függvény—automatikus végrehajtás" című résznél, a 219. oldalon.

Az *acad.lsp* fájl automatikus betöltése

Az *acad.lsp* fájl lehetővé teszi egy AutoLISP rutinokból álló könyvtár betöltését az AutoCAD minden egyes elindítása alkalmával.

Valahányszor egy rajzhoz hozzákezd, az AutoCAD egy *acad.lsp* nevű fájlt keres a könyvtárútvonalán, és ha megtalálja, akkor tartalmát a memóriába tölti.

Az *acad.lsp* fájl tartalmazhatja egy vagy több rutin tényleges programját, illetve csupán load függvényhívások sorozatát. Ez utóbbi módszer előnyösebb, mivel könnyebb módosítani. Amennyiben a következő kódot ASCII fájlként, *acad.lsp* néven elmenti, a *myapp1.lsp*, *build.lsp* és *counter.lsp* fájlokat az AutoCAD indulásakor mindig betölti.

```
(load "myapp1")
(load "build")
(load "counter")
```

Mivel az AutoCAD az *acad.lsp* fájlt a könyvtárútvonal sorrendjében keresi, Ön különböző *acad.lsp* fájlokat tarthat az egyes rajzkönytárakban. Ezáltal meghatározott AutoLISP rutinokat tölthet be a rajz vagy a feladat jellege szerint.

Ha az *acad.lsp* fájl betöltése közben AutoLISP hiba következik be, a fájl hátralévő részét az AutoCAD már nem tölti be. Általában az okoz ilyen hibát, ha az *acad.lsp* fájlban betöltésre kijelölt fájlok nem léteznek, vagy nem találhatók a könyvtárútvonalon. Erre az esetre gondolva használhatja a load függvény *hibaeset* argumentumát. Az iménti példában szereplő *acad.lsp* fájl a következőképpen írható át (a *hibaeset* argumentumot is belevéve):

```
(princ (load "myapp1" "\nMYAPP1.LSP betöltése
sikertelen."))
(princ (load "build" "\nBUILD.LSP betöltése
sikertelen."))
(princ (load "counter" "\nCOUNTER.LSP betöltése
sikertelen."))
(princ)
```

Amennyiben a load függvény végrehajtása sikeres, akkor a fájl utolsó kifejezésének az értékével tér vissza (rendszerint az utoljára definiált függvény nevével, vagy a függvény használatára vonatkozó üzenettel). Hiba esetén viszont a *hibaeset* argumentum értékét adja vissza. Ebben a példában a load függvény visszatérési értékét a princ függvénynek adjuk át, így ez az érték megjelenik a képernyő parancssorterületén. Amennyiben például a *ddtype.lsp* fájl betöltése kapcsán jelentkezik hiba, a **princ** függvény a következő üzenetet írja ki, majd az AutoCAD folytatja a további két fájl betöltését.

MYAPP1.LSP fájl betöltése sikertelen

Megjegyzés Az *acad.lsp* fájlban, illetve az *.mnl* fájlban a command függvényt csak a **defun** utasításon belül használhatja. Az S::STARTUP függvénnyel lehet olyan parancsokat definiálni, amelyek kiadására a rajzszerkesztés megkezdésekor azonnal szüksége van.

Az *.mnl* fájl automatikus betöltése

Az AutoCAD által automatikusan betöltött fájlok másik fajtája az aktuális menüfájlhoz társuló *.mnl* kiterjesztésű fájl. Amikor az AutoCAD betölt egy menüfájlt, egyúttal egy azonos nevű, de *.mnl* kiterjesztésű fájlt is keres, és ha talál ilyet, akkor annak tartalmát a memóriába tölti.

Ez a módszer biztosítja azt, hogy a menü megfelelő működéséhez szükséges AutoLISP függvények mindig betöltésre kerüljenek. A szabvány AutoCAD menü, az *acad.mnu* például az *acad.mnl* fájl hibátlan betöltésén alapszik. Ez a fájl tartalmazza a menüben használt számos AutoLISP függvény definícióját. Az *.mnl* fájl betöltésére azt követően kerül sor, hogy az *acad.lsp* betöltése befejeződött.

Megjegyzés Ha egy menüfájlt az AutoLISP **command** függvényének segítségével tölt be (a szintaxis hasonló a következő formához `(command "menu" "ujmenu")`), a hozzákapcsolt *.mnl* fájl addig nem töltődik be, míg az egész AutoLISP rutin le nem futott.

Például ha egy *ujmenu.mnu* nevű saját menüt hoz létre, amelynek helyes működéséhez három AutoLISP fájl, az *uj1.lsp*, *uj2.lsp* és *uj3.lsp* betöltésére van szükség, akkor készítsen egy *ujmenu.mnl* nevű ASCII szövegfájlt a következő tartalommal:

```
(load "új1")
(load "új2")
(load "új")
(princ "\nAz újmenü segédprogramjai ... Betöltve.")
(princ)
```

Ebben a példában a **princ** függvény meghívása használható státuszüzenetek megjelenítésére. A **princ** első hívásakor a parancssorban a következő üzenet jelenik meg:

Az újmenü segédprogramjai... Betöltve.

A **princ** második meghívása egy AutoLISP függvény "csendben" történő befejezésének a módszerét mutatja be.A második hívás hiányában az üzenet kétszer jelenne meg. Amint azt az előzőekben már említettük, a load függvény hívásakor a *hibaeset* argumentum megadásával különleges óvintézkedéseket is tehet.

Az automatikus betöltés parancsa

Amikor a fent leírt módszerek egyikével automatikusan betölt egy parancsot, a parancs definíciója a parancs tényleges használatakor lefoglalja a rendszermemória egy részét. Az AutoLISP **autoload** függvénye anélkül tesz elérhetővé egy parancsot, hogy az egész rutint a memóriába töltené. Ha az *acad.lsp* fájlhoz hozzáadja az alábbi kódot, az automatikusan betölti a CMD1, CMD2 és CMD3 parancsokat a *cmds.lsp* fájlból és a NEWCMD parancsot a *newcmd.lsp* fájlból.

```
(autoload "CMDS" '("CMD1" "CMD2" "CMD3"))
(autoload "NEWCMD" '("NEWCMD"))
```

Amint a Parancs prompthoz első alkalommal billentyűzi be egy automatikusan betöltődött parancs nevét, az AutoLISP a kapcsolódó fájlból betölti a teljes parancsdefiníciót. További információt talál az autoload függvényről az „autoload" címszót alatt a 13. fejezetben. Az AutoLISP az ADS és ARX alkalmazások számára is nyújt automatikus betöltési lehetőséget (további tudnivalók az „autoload" és „autoarxload" függvényekről a 13. fejezetben).

ADS—AutoCAD fejlesztői rendszer

Az AutoCAD fejlesztői rendszer (AutoCAD Development System, ADS) egy C nyelvű programozói környezet, amelyet AutoCAD alkalmazások fejlesztésére használhat. A C nyelven írt, ADS alapú alkalmazásokat az AutoCAD ugyanúgy kezeli, mint az AutoLISP nyelven megírt függvényeket. Egy adott ADS alkalmazás nem különálló programként jelenik meg, hanem mint *külső függvények* csoportja, amelyet az AutoLISP értelmező tölt be, illetve hív meg.

Mivel egy alkalmazás ADS vagyAutoLISP nyelven megírva funkcionálisan nagyrészt egyenértékű, ezért a fejlesztőtől függ, hogy melyiket választja. Az ADS alkalmazások sebesség és memóriafoglalás szempontjából általában hatékonyabbak, ezenkívül olyan szolgáltatásokat is nyújtanak, amelyeket az AutoLISP nem — például az operációs rendszer vagy a hardver közvetlen elérését. Ezért az ADS azon alkalmazások esetében kifejezetten előnyös, ahol jelentős a számítási igény, illetve az operációs rendszerrel való együttműködés. Másrészről azonban fejlesztésük és karbantartásuk időigényesebb és költségesebb lehet. Akárcsak magából az AutoCAD programból, az ADS alkalmazásokból is minden egyes számítógép-platformhoz más-más futtatható programváltozatot kell készíteni, és a fejlesztőeszközöket is minden egyes platformhoz külön meg kell vásárolni. Az AutoLISP előnyösebb a kezdeti kísérletezés időszakában, valamint olyan alkalmazások esetében, amikor a karbantartási és fejlesztési költségek lényegesebbek, mint a teljesítmény. Általában ez a helyzet azoknál az alkalmazásoknál, amelyek nagymértékben interaktívak, ugyanis ezek nagyban függnek a felhasználó által adott válasz sebességétől. Ugyanez igaz azokra az alkalmazásokra, amelyeknek sokat kell az AutoCAD reagálására várniuk.

Megjegyzés Amikor az AutoCAD az ads_command() függvénytől kapja a parancsot, a PICKADD és PICKAUTO rendszerváltozók feltételezett értéke 1, illetve 0. Ez biztosítja a kompatibilitást az AutoCAD előző verzióival, és megkönnyíti az alkalmazás készítőjének a dolgát (mivel nem kell ellenőriznie ezeknek a változóknak a beállítását).

Az ADS programok fejlesztésének feltételei

Amennyiben ADS alkalmazásokat szeretne fejleszteni vagy módosítani, akkor alaposan ismernie kell a C nyelvet, a professzionális szoftverfejlesztési technikákat, valamint magát az AutoCAD rendszert. Ha Ön gyakorlott programozó, de új az AutoCAD világában, illetve ha nem jártas a C nyelvben, tanácsos az ADS használata előtt megismerkednie az AutoLISP nyelvvel.

Az ADS környezet definiálása hasonlóan a szabvány C könyvtárakhoz, könyvtárakkal és header-fájlokkal történik. Ezek szinte az összes olyan *platform* (számítógép és operációs rendszer) esetében elérhetők, amelyeket az AutoCAD jelenleg támogat. Ennélfogva a legtöbb ADS alkalmazás — a forráskód szintjén — hordozható a különböző platformok között. A fejlesztő az ADS alkalmazás elkészítésére az adott környezetnek megfelelő fordító- (compiler) és szerkesztő- (link editor) programokat veheti igénybe. Az ADS könyvtárfüggvényei amellett, hogy biztosítják a hordozhatóságot, megkímélik a fejlesztőt attól, hogy különböző rendszerközeli részleteket kelljen megértenie, és azokkal dolgoznia. Az ADS függvénykönyvtár biztosítja az összes olyan eszközt, amely az AutoLISP interpreterrel és az AutoCAD rendszerrel való kommunikációhoz szükséges.

Vagy az operációs rendszerének, vagy a fordítócsomagjának tartalmaznia kell olyan szerkesztőprogramot is, amely képes ahhoz a hardverplatformhoz tartozó objektumfájlok kezelésére, amelyen az ADS alkalmazást futtatni szándékozik. Az ADS alkalmazást csatolni kell az ADS programkönyvtárhoz. Néhány platform esetében az ADS könyvtáron kívül egyedi kiegészítő modulok is szükségesek.

Az AutoCAD Release 12 verziójában megírt és lefordított ADS alkalmazások futása nem garantálható a Release 13 verzió alatt. Néhány alkalmazás lefuthat, de a korrekt működés érdekében ajánlatos az alkalmazásait újraszerkeszteni és újrafordítani.

ADS alkalmazások használata

Számos AutoCAD parancsot ADS alkalmazások definiálnak. A mintaprogramok az *ads* alkönyvtárban találhatók, mind lefordított, mind forrásnyelvi változatban. A lefordított programok a DOS verzió esetében *.exp*, illetve *.exe* kiterjesztésűek, míg a forrásnyelvi ADS programok ASCII szövegfájlok *.c* kiterjesztéssel.

EgyADS alkalmazás betöltése hasonló egy AutoLISP fájl betöltéséhez. Az xload AutoLISP függvényt használhatja arra, hogy az AutoCAD parancssorából utasítást adjon egy ADS alkalmazás betöltésére. Az xload függvény szintaxisa közel azonos az AutoLISP fájlok esetében használt **load** függvényével. Ha az xload függvény sikeresen betöltötte

az ADS programot, akkor a program nevével tér vissza. Az **xload** függvény szintaxisa a következő:

```
(xload fájlnév [hibaeset])
```

Az **xload** függvény két argumentuma a *fájlnév* és a *hibaeset*. Akárcsak a **load** függvény esetében, itt is meg kell adni a *fájlnév* argumentumot. Ennek egy betöltendő ADS programfájl nevét vagy útvonal/fájlnév meghatározását kell tartalmaznia. A *hibaeset* argumentum opcionális, és általában nem használatos a parancssorból történő **xload** függvényhíváskor. A következő példa azt mutatja be, hogyan lehet a parancssorból betölteni a *geomcal.exp* ADS programfájlt, amely egy geometriai számológépnek a parancssorból történő elérését teszi lehetővé:

Parancs: **(xload "geomcal")**

Akárcsak az AutoLISP fájlok esetében, a megadott fájl keresése itt is az AutoCAD könyvtárútvonal mentén történik. Ne feledje, hogy a teljes elérési útvonal megadásakor normál törtjelet (/) vagy két visszahajló törtjelet (\\ ) kell használnia a könyvtárak tagolására.

Amennyiben egy előzőleg már betöltött programot próbál újra betölteni, akkor az alábbi üzenet fog megjelenni (ahol *xxx* az alkalmazás neve).

"*xxx*" alkalmazás már be van töltve

Az ADS alkalmazások rendszermemóriát foglalnak le. Ha már nincs szüksége egy alkalmazásra és ki akarja azt törölni a memóriából, használhatja az AutoLISP **xunload** függvényét. Az alábbi példa kitörli a *geomcal* alkalmazást.

Parancs: **(xunload "geomcal")**

A xunload függvény nemcsak kitörli az alkalmazást a memóriából, hanem a hozzá kapcsolódó parancsdefiníciókat is eltávolítja. Ha az xunload használatával kitörölte a Geomcal alkalmazást, és később még ugyanabban a rajzban kiadja a KAL parancsot, akkor az AutoCAD Ismeretlen parancs üzenettel válaszol. Ha ez megtörténik, az alkalmazást manuálisan újra be kell töltenie az xload függvény használatával.

Az ADS alkalmazások automatikus betöltése

Az *acad.ads* azon ADS programfájlok felsorolását tartalmazza, amelyeket az AutoCAD induláskor automatikusan betölt. Ez a fájl olyan editorral vagy szövegszerkesztővel szerkeszthető, amely alkalmas ASCII fájlok létrehozására. Ezt a fájlt tetszése szerint átalakíthatja (hozzáadhat vagy törölhet a tartalmából), biztosítva a saját céljainak megfelelő ADS programok elérését.

Megjegyzés Az *acad.ads* fájl eredeti tartalmának törlése esetén bizonyos AutoCAD parancsok nem fognak működni.

Mivel az AutoCAD az *acad.ads* fájlt a könyvtárútvonal sorrendjében keresi, Ön különböző *acad.ads* fájlokat tarthat az egyes rajzkönyvtárakban. Ily módon meghatározott ADS programokat tehet elérhetővé a rajz (vagy a feladat) jellege szerint. A háromdimenziós modellek rajzait tarthatja az *acad/munka/3d_rajz* nevü könyvtárban. Feltéve, hogy ez az aktuális könyvtár, az *acad.ads* fájlt bemásolhatja ebbe a könyvtárba, és a következőképpen módosíthatja:

```
acadapp                 az acad.ads fájl alapértelmezésbeli tartalma

geom3d
geomcal                 lehetséges kiegészítések—3D alkalmazások
```

Amennyiben ezt az új *acad.ads* fájlt elhelyezi az *acad/munka/3d_rajz* könyvtárban, és ebből a könyvtárból indítja el az AutoCAD programot, akkor ezen új ADS programok betöltésre kerülnek, és elérhetővé válnak az AutoCAD parancssorából. Amennyiben a prgramot egy másik könyvtárból indítja, amely nem tartalmaz *acad.ads* fájlt, akkor az AutoCAD az alapértelmezés szerinti *acad.ads* fájlt fogja betölteni, mivel az eredeti *acad.ads* fájl továbbra is megtalálható az AutoCAD programfájljaival azonos könyvtárban.

Lehetősége van arra is, hogy az *.mnl* fájlban elhelyezett **xload** függvénnyel töltsön be ADS programokat. Így biztosítható, hogy a menüfájl betöltésével együtt a menü helyes működéséhez szükséges ADS programok is betöltésre kerüljenek.

Az automatikus betöltési lehetőséget kihasználhatja az ADS által definiált AutoCAD parancsok esetében is. További tudnivaló "Az automatikus betöltés parancsa" című résznél, a 164. oldalon és a „autoarxload" függvénynél (13. fejezet).

ARX—AutoCAD futásideje alatti kiterjesztése

Az ARX (AutoCAD Runtime Extension, vagyis az AutoCAD futásideje alatti kiterjesztése) egy új, az AutoCAD alkalmazások fejlesztésére szolgáló fordítást igénylő programnyelveket alkalmazó programkörnyezet. Az ARX környezet jelenleg az ADS könyvtárat támogatja, és a jövőben támogat majd sok egyéb könyvtárat is.

Az ARX környezet közvetlenül kommunikál az AutoCAD programmal, míg az ADS az AutoLISP értelmezőn keresztül teszi ezt. A nagyobb teljesítmény érdekében az ARX alkalmazások ugyanabban a folyamatban és memóriatartományban működnek, mint az AutoCAD.

Ezen kívül az API-k az ARX környezetből sokkal hatékonyabban exportálhatók, mint az ADS programkörnyezetből.

ARX alkalmazásokat az AutoLISP arxload függvényével tölthet be. Az arxload függvény szintaxisa megegyezik a korábban tárgyalt load és xload függvényekével. Az AutoCAD Render és ASE parancsait az ARX alkalmazások definiálják. Mivel minden Render és ASE parancs automatikus betöltésű, ezen parancsok egyikének kiadásakor a megfelelő ARX alkalmazás töltődik be.

Ha már nincs szüksége egy ARX alkalmazás által definiált parancsokra, akkor törölheti az alkalmazást az arxunload függvény segítségével. Az arxunload függvény nemcsak kitörli az alkalmazást a memóriából, hanem a hozzá kapcsolódó parancsdefiníciókat is eltávolítja. Ha az arxunload használatával kitörölte a Render alkalmazást, és később még ugyanabban a rajzban kiadja a RENDER parancsot, akkor az AutoCAD Ismeretlen parancs üzenettel válaszol. Ennek elkerülésére a RENDERTÖRÖL és ASEUNLOAD parancsok használhatók. Ezek az acadr13.lsp fájlban definiált parancsok eltávolítják az ARX alkalmazást a memóriából, majd automatikusan betöltik az összes kapcsolódó parancsot.

Az automatikus betöltési lehetőséget kihasználhatja az ARX által definiált AutoCAD parancsok esetében is. További tudnivaló az "Az automatikus betöltés parancsa" című résznél, a 164. oldalon és az „autoarxload" függvénynél (13. fejezet).

Egy .mnl fájlból az arxload függvény segítségével tölthet be ARX alkalmazásokat. Ez biztosítja, hogy a menüfájl betöltésekor a menü megfelelő működéséhez szükséges ARX alkalmazás is betöltődjön.

II. RÉSZ
AZ AUTOLISP ALAPJAI

TARTALOM

II. RÉSZ AUTOLISP REFERENCIA 187

7. FEJEZET ■ AZ AUTOLISP ALAPJAI 189

AutoLISP kifejezések 190
AutoLISP adattípusok 191
Szubrutinok és külső szubrutinok 191
Egész típusú számok 191
Valós típusú számok 192
Karakterláncok 192
Listák 192
Kiválasztási halmazok 192
Rajzelemnevek 193
Fájlleírók 193
Szimbólumok és változók 193
AutoLISP függvények szintaxisa 194
AutoLISP programfájlok 194
Megjegyzések 195
Tabulálás és igazítás 196
AutoLISP változók 196
Változók használata a parancssorban 197
Előre definiált változók 198
Számok kezelése 198
Karakterlánc-kezelés 199
Vezérlőkarakterek a karakterláncokban 201
Összehasonlítás globális karakterekkel 204
Relációs függvények 205
Listakezelés 206
Pontlisták 207

Kapcsolt értékpárok 209
Szimbólum- és függvénykezelés 210
C:*XXX* függvények 211
Parancsok készítése 212
AutoCAD parancsok átdefiniálása 213
Lokális szimbólumok a függvényekben 214
Argumentummal rendelkező függvények 215
Hibakezelés 216
Alkalmazások kezelése 218
AutoLISP alkalmazások betöltése 218
S::STARTUP függvény—automatikus végrehajtás 219
Betöltési módszerek 219
ADS és ARX alkalmazások betöltése 220

8. FEJEZET ■ ÁLTALÁNOS SEGÉDFÜGGVÉNYEK 223

Lekérdező és parancsfüggvények 224
Parancsok meghívása a command függvénnyel 224
Nemzeti (honosított) nyelv-támogatás 225
Várakozás a felhasználói adatbevitelre 225
Kijelölt pontok átadása az AutoCAD parancsoknak 226
Rendszer- és környezeti változók 226
Konfigurációvezérlés 227
Képernyővezérlés 227
Üzenetek megjelenítése a parancssorban 227
Menük vezérlése 228
A grafikus és szöveges ablakok vezérlése 231
Alacsonyszintű grafika vezérlése 232
Felhasználói bevitel kérése 232
A get*xxx* függvények 233
A felhasználói bevitelre szolgáló függvények feltételeinek szabályozása 235
A felhasználói bevitelre szolgáló függvények bemeneti opciói 236
Kulcsszó opciók 236
Tetszőleges bevitel a billentyűzetről 237
A bevitel érvényessége 238
Geometriai segédfüggvények 238
Tárgyraszter 239
A szöveg kiterjedése 239
Konverziók 243
Karakterlánc-konverziók 243
Szögkonverziók 246
ASCII kódok konverziója 246
Mértékegységek átalakítása 248

Konverzió hüvelykről méterre 249
A mértékegység-definíciós fájl 249
Koordináta-rendszerek közötti transzformációk 251
Pontok transzformációja 254
Fájlkezelés 254
Fájlkeresés 254
Súgófájlok elérése 255
Az eszközök elérése és vezérlése 256
Felhasználói adatbevitel elérése 256
A digitalizáló táblák kalibrálása 257
Az ASE, az AutoLISP felhasználói felülete 259
Adattípusok 259
Adatbázis objektumnév —*obj_name* 259
Csatolási útvonalnév—*lp_name* 260
Hivatkozás adatbázis-objektumra — *obj_path* 260
Oszlopnév—*column_name* 260
Kulcsérték—*key* 261
Csatolás azonosító—*link_id* 261
Csatolás kiválasztás—*link_sel* 261
Végrehajtás 261
Általánosító megjegyzések 262
Hibadiagnosztika 262

9. FEJEZET ■ KIVÁLASZTÁSI HALMAZOKAT, OBJEKTUMOKAT ÉS SZIMBÓLUMTÁBLÁKAT KEZELŐ FÜGGVÉNYEK 263

Kiválasztási halmazok kezelése 264
Kiválasztási halmazok szűrőlistái 266
Globális karakterminták a szűrőlistákban 268
Bővített adatok szűrése 269
Relációvizsgálat 269
Szűrővizsgálatok logikai csoportosítása 270
Műveletek kiválasztási halmazokkal 271
Objektumkezelés 272
Rajzelemnév függvények 273
Rajzelem-sorszámok és használatuk 274
Rajzelemek összefüggései és koordináta-transzformációs adatok 275
Rajzelem-elérési függvények 278
Rajzelemadatokat kezelő függvények 279
Névtelen blokkok 283
Összetett rajzelemek létrehozása 283
Rajzelemadatokat kezelő függvények és a grafikus képernyő 285
Görbe és spline vonalláncok feldolgozása 286
Szimbólumtábla-objektumok 287

Szótárobjektumok 288
Bővített adatok—Xdata 289
Bővített adatok szervezése 290
Alkalmazások bejegyzése 293
Bővített adatok lekérdezése 294
Bővített adatok hozzákapcsolása egy rajzelemhez 295
A bővített adatok által használt memória kezelése 296
Sorszámok a bővített adatokban 296
Hozzáférés szimbólumtáblákhoz és szótárakhoz 298
Szimbólumtáblák 298
Szótárak 300

10. FEJEZET ■ PÁRBESZÉDPANELEK KEZELÉSE 301

Párbeszédpanelek megnyitása és bezárása 302
Zóna- és attribútumkezelés 304
Zónakifejezések és visszajelzések 305
Akciókifejezések 306
Visszajelzési indítékok 307
Alapértelmezés szerinti és DCL akciók 309
Zónák kezelése 309
Zónamódok és zónaértékek inicializálása 309
Állapotok és értékek megváltoztatása a visszajelzés során 310
Rádiócsoportok kezelése 311
Csúszkák kezelése 312
Szerkesztőablakok kezelése 313
Párbeszédpanelek egymásba ágyazása 313
Párbeszédpanelek elrejtése 314
Listaablakok és legördülő listák 317
Ábrazónák és nyomógombok 319
Ábrák létrehozása 320
Ábra-nyomógombok kezelése 322
Alkalmazás-specifikus adatok 322
A párbeszédpanel-függvények összefoglalása 324
Függvényhívások sorrendje 324
Párbeszédpanel-minta blokk definiálásához 325

11. FEJEZET ■ AUTOLISP GYAKORLÓ PÉLDA 327

A kitűzött cél 328
Lássunk munkához! 328
Adatbevitel 330
Tájékozódás 333
A burkolólapok megrajzolása 334
A parancs csatolása az AutoCAD rendszerhez 336
Kiegészítés párbeszédpaneles felülettel 338

A DCL fájl—*dpkj.dcl* 339
Párbeszédpaneleket kezelő AutoLISP függvények—*dpkj.lsp* 340

12. FEJEZET ■ AZ AUTOLISP FÜGGVÉNYEK ÁTTEKINTÉSE 345

Alapfüggvények 346
Aritmetikai függvények 346
Karakterlánc-kezelő függvények 348
Egyenlőségi és feltételes függvények 349
Listakezelő függvények 350
Szimbólumkezelő függvények 352
Függvénykezelő függvények 353
Hibakezelő függvények 353
Alkalmazáskezelő függvények 354
Segédfüggvények 355
Lekérdező- és parancs függvények 355
Megjelenítésvezérlő függvények 356
Felhasználói bevitelre szolgáló függvények 357
Geometriai függvények 359
Konverziós függvények 360
Fájlkezelő függvények 361
Eszközelérési függvények 361
Az ASE felület függvényei 362
ASE adminisztratív függvények 362
ASE csatoláselérési függvények 363
ASE csatolás kiválasztás-kezelő függvények 363
ASE segédfüggvények 365
ASE hibakezelő függvények 365
Kiválasztási halmaz, objektum és szimbólumtábla függvények 366
Kiválasztási halmaz kezelő függvények 366
Objektumkezelő függvények 367
Bővített adatokat kezelő függvények 367
Szimbólumtábla- és szótárkezelő függvények 368
Programozható párbeszédpanel függvények 369
Párbeszédpanel nyitó és záró függvények 369
Zóna- és attribútumkezelő függvények 369
Listaablakokat és legördülő listákat kezelő függvények 370
Ábrazóna-kezelő fügvények 370
Alkalmazás-specifikus adatokat kezelő függvények 371
Memóriakezelő függvények 371

13. FEJEZET ■ AUTOLISP FÜGGVÉNYEK KATALÓGUSA 373
+ (összeadás) 374
– (kivonás) 374
* (szorzás) 374
/ (osztás) 375
= (egyenlő) 375
/= (nem egyenlő) 376
< (kisebb, mint) 376
<= (kisebb vagy egyenlő) 376
> (nagyobb, mint) 377
>= (nagyobb vagy egyenlő) 377
~ (bitenkénti NOT) 377
1+ (növelés) 378
1– (csökkentés) 378
abs 378
acad_colordlg 378
acad_helpdlg 379
acad_strlsort 379
action_tile 380
add_list 380
ads 381
alert 381
alloc 382
and 382
angle 382
angtof 383
angtos 384
append 385
apply 385
arx 385
arxload 386
arxunload 386
ase_docmp 387
ase_docurrent 387
ase_dolist 388
ase_dopathcode 389
ase_dopathmake 390
ase_dopathname 390
ase_dostatus 391
ase_errcode 392
ase_errdsc 392
ase_errmsg 392
ase_errqty 393
ase_linkcreate 393
ase_linkget 395
ase_linkremove 395
ase_linkupdate 396
ase_lpcreate 396
ase_lperase 397

ase_lpisupdatable 398
ase_lpkey 398
ase_lplist 398
ase_lppath 399
ase_lprename 399
ase_lsadd 399
ase_lscmp 400
ase_lscopy 400
ase_lscreate 400
ase_lsdel 402
ase_lsentsel 402
ase_lserase 403
ase_lsfree 403
ase_lsget 403
ase_lsintersect 404
ase_lsintersectfilter 404
ase_lsisupdatable 405
ase_lslpnames 405
ase_lsmemb 406
ase_lsqty 406
ase_lssubtract 406
ase_lsunite 407
ase_lsxnames 407
ase_version 408
ascii 408
assoc 408
atan 409
atof 409
atoi 410
atom 410
atoms-family 410
autoarxload 411
autoload 411
autoxload 412
Boole 412
boundp 414
car and cdr 414
chr 416
client_data_tile 416
close 416
command 416
cond 419
cons 420
cos 420
cvunit 420
defun 421
dictnext 423
dictsearch 423
dimx_tile és

dimy_tile 424
distance 424
distof 425
done_dialog 426
end_image 426
end_list 427
entdel 427
entget 427
entlast 429
entmake 430
entmod 433
entnext 434
entsel 435
entupd 436
eq 437
equal 437
error 438
eval 439
exit 439
exp 439
expand 440
expt 440
fill_image 440
findfile 441
fix 442
float 442
foreach 442
gc 443
gcd 443
get_attr 443
get_tile 444
getangle 444
getcfg 445
getcorner 445
getdist 446
getenv 446
getfiled 447
getint 449
getkword 450
getorient 451
getpoint 451
getreal 452
getstring 452
getvar 453
graphscr 454
grclear 454
grdraw 454
grread 455
grtext 458

grvecs 459
handent 460
if 461
help 461
initget 462
inters 466
itoa 466
lambda 467
last 467
length 468
list 468
listp 468
load_dialog 469
load 469
log 470
logand 470
logior 471
lsh 471
mapcar 471
max 472
mem 472
member 473
menucmd 473
min 475
minusp 475
mode_tile 476
namedobjdict 476
nentsel 476
nentselp 479
new_dialog 480
not 481
nth 481
null 481
numberp 482
open 482
or 484
osnap 484
polar 484
prin1 485
princ 486
print 486
progn 486
prompt 487
quit 487
quote 487
read 488
read-char 488
read-line 489
redraw 489

regapp 490
rem 491
repeat 491
reverse 491
rtos 492
set 493
set_tile 493
setcfg 493
setfunhelp 494
setq 495
setvar 495
sin 496
slide_image 497
snvalid 497
sqrt 497
ssadd 498
ssdel 498
ssget 499
Kiválasztási halmazok szűrői 501
sslength 504
ssmemb 504
ssname 504
startapp 505
start_dialog 505
start_image 506
start_list 506
strcase 507
strcat 507
strlen 508
subst 508
substr 508
tablet 509
tblnext 510
tblobjname 512
tblsearch 512
term_dialog 513
terpri 513
textbox 513
textpage 514
textscr 514
trace 514
trans 515
type 517
unload_dialog 518
untrace 518
vector_image 519
ver 519
vmon 520
vports 520

wcmatch 521
while 523
write-char 524
write-line 524
xdroom 525
xdsize 525
xload 526
xunload 527
zerop 527

14. FEJEZET ■ A KÜLSŐLEG DEFINIÁLT PARANCSOK ÉS RENDSZERVÁLTOZÓK HASZNÁLATA 529

3DSBE 530
3DSKI 531
ANYAGTÁR 532
ASEADMIN 532
ASEEXPORT 533
ASELINKS 533
ASEROWS 533
ASESELECT 534
ASESQLED 534
FÉNY 534
A—Ambient (Szórt háttérvilágítás) 535
D—Delete (Töröl) 535
L—List (Listáz) 535
M—Modify (Módosít) 536
ND—New Distant Light (Új távoli fényforrás) 537
NP—New Point Light (Új pontszerű fényforrás) 538
NS—New Spotlight (Új reflektor) 539
R—Rename (Átnevez) 540
FORGAT3D 540
GIFBE 541
ILLESZT 541
JELENET 542
D—Delete (Töröl) 542
L—List (Listáz) 542
M—Modify (Módosít) 543
N—New (Új) 544
R—Rename (Átnevez) 545
S—Set (Beállít) 545
KAL 545
KÉPMENTŐ 546
KÉPÚJRA 547
PCXBE 548
PSBE 548
PSKÉP 549
PSKITÖLT 549

RANY 550
A—Attach (Hozzárendel) 550
C—Copy (Másol) 551
D—Detach (Leválaszt) 551
L—List (Listáz) 552
M—Modify (Módosít) 552
N—New (Új) 552
RBEÁLL 554
ACMAP—AutoCAD paletta 554
DEST—Kimenet 555
ICON—Ikon 555
STYPE—Rendertípus 555
SELECT—Kiválaszt 556
TOGGLE—Kapcsoló 556
RENDER 557
RENDERKÉP 557
RIASPECT 558
RIBACKG 558
RIEDGE 559
RIGAMUT 559
RIGREY 559
RITHRESH 560
RKONFIG 560
RSTAT 560
TIFFBE 561
TÜKRÖZ3D 561
VISLKONV 562

15. FEJEZET ■ MEMÓRIAKEZELÉS 563

Csomóponti terület 564
Csomóponti terület felszabadítása 564
Technikai megjegyzések 565
Karakterlánc-terület 565
Szimbólumok tárolása 565
Manuális allokáció 566
Virtuális függvénylapozás 567

16. FEJEZET ■ ALKALMAZÁS-SPECIFIKUS DXF CSOPORTKÓDOK 571

Csoportkódok emelkedő számsorrendben 572
Grafikus rajzelemek 575
Általános csoportkódok 575
Rajzelemtípus szerinti csoportkódok 576
A méretezési stílus felülírása 588
Nem-grafikus rajzelemek 588

Általános csoportkódok 588
Szótár csoportkódok 589
Blokk és szimbólumtábla csoportkódok 590

17. FEJEZET ■ AUTOLISP HIBAKÓDOK ÉS HIBAÜZENETEK 597

Hibakódok 598
Hibaüzenetek 601
Felhasználói programok hibái 602
Belső hibák 608

II. RÉSZ
AUTOLISP REFERENCIA

Az LISP programozási nyelv egyik változata, az AutoLISP, az AutoCAD programcsomag integrált része. Az AutoLISP olyan hatékony, magasszintű, a grafikus alkalmazásokhoz illeszkedő nyelv, melynek használatával makróprogramok és függvények írhatók. Az AutoLISP igen rugalmas, használata egyszerű.

Az *AutoLISP Referencia* alapvető útmutató az AutoLISP programozási nyelvhez. Az általános LISP nyelv részeként, az AutoLISP megtartja annak leírási módját és egyezményes szabályait; ugyanakkor számos, az AutoCAD programhoz kapcsolódó új függvényt tartalmaz. Ha többet szeretne tudni az AutoLISP programozási nyelvről, számos könyvet olvashat a témában. Ajánljuk: Winston, Horn *LISP* című könyvét (második kiadás), valamint Tony Hasemer *Looking at LISP* című munkáját (mindkettő az Addison-Wesley kiadó gondozásában jelent meg).

Számos AutoLISP programot, beleértve az itt leírt példaprogramokat is, az AutoCAD *sample* könyvtárában is megtalálhat. Néhányat shareware programként, vagy más forgalmazóktól szerezhet be. Mivel az AutoLISP parancsait a parancssorba is bebillentyűzheti, könnyen megy majd a tanulás. Ha pedig elsajátította az AutoLISP nyelvet, növelheti az AutoCAD parancsok számát.

Ez a referencia az AutoLISP függvények használatára és AutoLISP alkalmazások írására összpontosít. A 6., "Programozási felületek" című fejezet általános áttekintést nyújt az AutoLISP nyelvről, valamint már létező AutoLISP alkalmazások betöltéséről.

7

AZ AUTOLISP ALAPJAI

Ez a fejezet az AutoLISP programozási nyelv alapelveit ismerteti. Mivel az AutoLISP kód nem fordítódik le, a parancssorban megadott kód eredménye rögtön megjelenik. A fejezetben található példákat úgy alakítottuk ki, hogy a kezdők is könnyen beírhassák őket a parancssorba.

AutoLISP kifejezések

Egy Parancs prompthoz beírt szöveget az AutoLISP úgy értelmez, hogy a szöveg karaktereit összehasonlítja egy belső listával, mely az érvényes parancsszavakat tartalmazza. Ha a szó, amit megadott egyezik a listán találhatók valamelyikével, kiértékeli azt a parancs-definíciót, mely az adott utasítást végrehajtja. Amikor az AutoCAD AutoLISP kódot kap, az értelmező programnak továbbítja azt. Az AutoLISP értelmező programon belül található a kiértékelő. A kiértékelő beolvas egy kódsort, értelmezi azt, és visszaadja az eredményt. A kódnak AutoLISP formátumúnak kell lennie, melyet a program vagy egy fájlból, vagy a felhasználó által az AutoCAD parancssorban bevitt adatból olvas be.

Az AutoLISP kifejezések a következő formátumúak.

(függvény *argumentumok*)

Mindegyik kifejezés egy baloldali zárójellel kezdődik, és egy függvény nevéből, valamint annak opcionális argumentumlistájából áll (melyek mindegyike lehet maga is egy kifejezés). A kifejezés a jobb oldali zárójellel végződik. Minden kifejezés egy olyan értéket ad vissza, amelyet egy azt körülvevő kifejezés felhasználhat; amennyiben nem veszi körül másik kifejezés, akkor az AutoLISP az eredményt az AutoCAD parancssorának adja át.

A következő kódpélda három függvényt foglal magába.

```
(füg1 (füg2 argumentumok)(füg3 argumentumok))
```

A **füg1** nevű függvény két argumentumot, míg a másik kettő egyet-egyet tartalmaz. A **füg1** és a **füg3** függvények a **füg1** függvénybe vannak beágyazva, így visszatérési értékei egyben a **füg1** argumentumai is. A **füg1** függvény ezt a két értéket értékeli ki, és a végeredményt a parancssorba írja.

Amikor az AutoCAD Parancs promptjába egy AutoLISP kifejezést ír, az AutoLISP kiértékeli a kifejezést, kijelzi az eredményt, és a Parancs prompt újra megjelenik. A következő példa a * (szorzás) függvény használatát mutatja be, mely argumentumként egy vagy több valós számot fogad el.

Parancs: **(* 2 27)**
54

Mivel ezt a kódot nem veszi körül kifejezés, az eredményt a parancssornak adja át.

Más kifejezésekbe ágyazott kifejezések az őket körülvevő kifejezéseknek adják át az eredményt. A következő példa a +

(összeadás) függvény eredményét használja a * (szorzás) függvény egyik argumentumaként.

Parancs: **(* 2 (+ 5 10))**
30

Ha nem megfelelő számú lezáró zárójelet tett ki, az AutoLISP a következő prompttal jelenik meg:

n>

Ahol az *n* egész szám jelzi, hogy a baloldali zárójeleknek hány szintje maradt lezáratlan. Amennyiben ez a prompt jelenik meg, akkor a kifejezés kiértékeléséhez *n* darab jobboldali zárójelet kell bebillentyűznie.

Parancs: **(* 2 (+ 5 10**
2>**))**
30

Gyakran előforduló hiba még a karakterláncot lezáró idézőjel (") elhagyása; ilyen esetben a jobb zárójeleket az értelmező úgy értékeli, mintha a karakterlánchoz tartoznának, ezért újabb zárójelek beírása sem változtat az *n* értékén. Ezen úgy segíthet, ha a függvényt a Ctrl+C vagy az Esc billentyűkkel megszakítja, majd helyesen újra bebillentyűzi.

AutoLISP adattípusok

Az AutoLISP kiértékelő a zárójelen belül szereplő programrészek adattípusától és sorrendjétől függően értékeli a kifejezéseket. Az AutoLISP tulajdonságainak kihasználása érdekében meg kell ismerkednie a különböző adattípusokkal és az azok közötti különbségekkel.

Szubrutinok és külső szubrutinok

A 13. fejezetben az "AutoLISP függvények katalógusa" című részben leírt függvények jó része beépített szubrutin, melyeknek neve *subrs*. A 14., "Külsőleg definiált parancsok és rendszerváltozók használata" című fejezetben leírt függvényeket külső, ADS és ARX alkalmazások definiálják, és *external subr* a nevük. Ezekben a szubrutin nevekben a program nem tesz különbséget nagy- és kisbetűk között.

A **`princ`** függvény egy szubrutin, az **`acad_strlsort`** függvény pedig egy külső szubrutin.

Egész típusú számok

Az egész típusú számok olyan értékek, melyek nem tartalmaznak tizedes jegyeket. Az AutoLISP egész számok 32 bites, előjeles számok +2147483648 és -2147483647 között. Bár az AutoLISP belsőleg 32

bites értékeket használ, az AutoCAD és az AutoLISP közötti kommunikációban csak 16 bites értékek vesznek részt. Ezért a +32767 -nél nagyobb illetve a -32768 -nál kisebb értékeket az AutoLISP nem adja át az AutoCAD programnak. Amikor egy AutoLISP kifejezésben egész számot használ, az *konstans* értékként jelenik meg. A 2, -56, és 1200196 mind érvényes AutoLISP egész számok.

Valós típusú számok

A valós típusú számok olyan értékek, melyek tizedes helyiértékeket is tartalmaznak. A -1 és +1 közötti értékek 0 -val kell, hogy kezdődjenek. A valós számok dupla pontosságú lebegőpontos számok, melyek minimum 14 helyiértékig pontosak; bár ebből az AutoCAD csak hatot jelenít meg. A valós számokat tudományos formában is kijelzi a program. Ez azt jelenti, hogy egy e vagy E jelet tesz a szám mögé, melyet a kitevő értéke követ (például a 0,0000041 ugyanannyi, mint 4,1e-6). Amikor egy AutoLISP kifejezésben valós számot használunk, az *konstans* értékként jelenik meg. A 3,1, 0,23 , -56, 123, és 21000000,0 mind érvényes AutoLISP valós számok.

Karakterláncok

A karakterláncok idézőjelek közötti karakterek sorozatai. A karakterláncon belüli visszahajló törtjel (\) jelet használva lehetőség van vezérlőkarakterek (vagy escape kódok) használatára. Egy karakterlánc maximum 132 karakter hosszú lehet. Amikor az AutoLISP kifejezésben karakterláncot használ, az *literal karakterlánc* vagy *karakterlánc konstans* értékként jelenik meg.

A következők érvényes karakterláncok: "karakterlánc 1" és "\nBillentyűzze be az első pontot:".

Listák

Az AutoLISP lista zárójelben lévő, szóközzel elválasztott, összetartozó értékek sorozata. A listák hatékony módszert jelentenek az összetartozó értékek tárolására. Az AutoCAD háromdimenziós pontokat három valós szám listájaként ad meg.

A következő példák listák: (1,0 1,0 1,0), ("ez" "az" "a másik") és (1 "EGY").

Kiválasztási halmazok

A kiválasztási halmazok egy vagy több objektumot (rajzelemet) tartalmazó csoportok. Az AutoLISP rutinokkal párbeszédes formában lehet a kiválasztási halmazokhoz objektumokat adni, vagy azokból objektumokat eltávolítani.

A következő példában az ssget függvény egy rajz összes objektumát tartalmazó kiválasztási halmazzal tér vissza.

Parancs: **(ssget "X")**
<Kiválasztási halmaz: 1>

Rajzelemnevek

Egy rajzelem neve olyan számozott címke, mely egy rajzon található objektumokhoz tartozik. A rajzelem-név valójában egy pointer, és egy olyan AutoCAD fájlra mutat, amelyben az AutoLISP megtalálja az objektum adatbázisát és vektorát (ha azok kijezésre kerülnek). Erre a címkére hivatkozhatunk AutoLISP függvényekben, így választhatók ki a felhasználandó objektumok. Belsőleg az AutoCAD az objektumokat rajzelemekként kezeli.

A következő példa az **`entlast`** függvény segítségével hívja be annak az objektumnak a nevét, amelyik legutoljára került a rajzba.

(entlast)
<elem-néve: 60000016>

Fájlleírók

A fájlleírók az AutoLISP által megnyitott fájlokhoz tartozó alfanumerikus címkék. Amikor az AutoLISP egy fájlt olvas vagy abba ír, a fájl címkéje el kell hogy legyen látva megjegyzésekkel.

A következő példa a *myinfo.dat* fájlt nyitja meg olvasásra.

(open "myinfo.dat" "r")
<Fájl: #34614>

Szimbólumok és változók

Az AutoLISP szimbólumokat használ az adatok azonosítására. A szimbólumnevekben kis- és nagybetűk, valamint alfanumerikus jelek sorozata és szimbólumjelek is használhatók, kivéve a ( ) . ' " ; szimbólumjeleket. Egy szimbólumnév nem tartalmazhat kizárólag számokat.

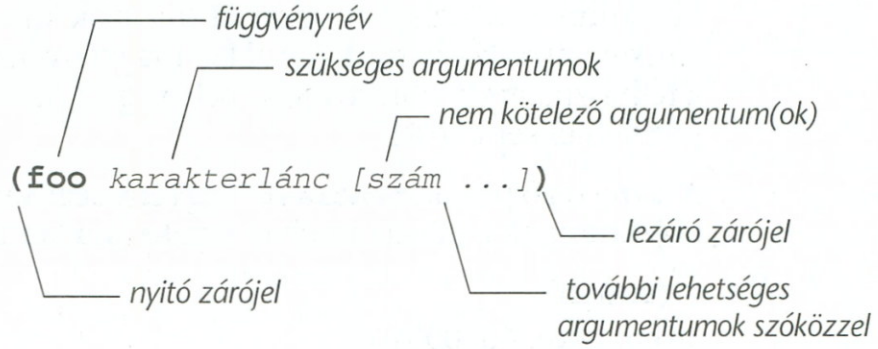
Az AutoLISP alkalmazások gyakorlatilag ugyanúgy tartalmaznak szimbólumokat, mint ahogy konstans értékeket is (például a karakterláncokat, a valós vagy az egész számokat). Ezért, a félreértések elkerülése végett, jelen útmutató a *szimbólum*-kifejezéssel olyan szimbólumnévre utal, amely statikus adatokat tartalmaz, mint például a beépített vagy a felhasználó által definiált függvények. A *változó* kifejezést olyan szimbólumnévnél használja a leírás, mely programadatokat tárol. A következő példa a setq függvényt használja

az str1 változó értékének "ez egy karakterlánc" karakterlánccal történő megadására.

```
. (setq str1 "ez egy karakterlánc")
"ez egy karakterlánc"
```

AutoLISP függvények szintaxisa

Ebben a kézikönyvben a következő szabályok érvényesek egy AutoLISP függvény leírására.



Az előző példában a foo függvénynek egy kötelező argumentuma van, ez egy karakterlánc; és egy választható, ez pedig szám típusú. További szám-argumentumok is megadhatók. Gyakran az argumentum neve utal a várt adat típusára. A következő táblázat a foo függvény érvényes és érvénytelen hívásait mutatja be.

A foo függvény érvényes és érvénytelen meghívásai

Érvényes meghívás	*Érvénytelen meghívás*
`(foo "catch")`	`(foo 44 13)`
`(foo "catch" 22)`	`(foo "fi" "foe" 44 13)`
`(foo "catch" 22 31)`	`(foo)`

AutoLISP programfájlok

Bár az AutoLISP-programot a parancssorban is megadhatja, a tesztelés és a hibakeresés sokkal könnyebb, ha azt egy fáljból tölti be. Így nem kell a programot minden alkalommal újból bebillentyűznie, amikor meg akarja változtatni. Az AutoLISP-programok általában ASCII szövegfájlok *.lsp* vagy *.mnl* kiterjesztéssel. Azonban bármilyen ASCII szövegfájlból betölthet egy AutoLISP-programot, ha megadja a load függvénynek a teljes fájlnevet. A fájlokban vagy a Parancs prompt után megadott AutoLISP kifejezések leírási módja azonos.

Megjegyzések

Az AutoLISP programfájlok gyakran tartalmaznak megjegyzéseket. Ezek a megjegyzések a programozónak és annak a felhasználónak is hasznosak, aki saját igényei szerint kívánja átalakítani az adott programot. A következő esetekben használhat megjegyzéseket.

- Megadhatja a program címét, szerzőjének nevét és a létrehozás dátumát.
- Használati utasítást mellékelhet egy rutinhoz.
- Magyarázatokat illeszthet egy rutinba.
- Jegyzeteket készíthet a hibakeresés alatt.
- A megjegyzésbe írhat látványos karaktereket.

A megjegyzéseket mindig pontosvessző (;) vezeti be, és sor végéig tartanak.

```
; Ez az egész sor egy megjegyzés
(setq area (* pi r r))  ; Kiszámolja a kör területét
```

A ;| ... |; jelek közé írt szöveget, a program nem veszi figyelembe; így a megjegyzések beékelhetők egyetlen programsorba, és több sor hosszúságúak is lehetnek. Íme egy *in-line* (soron belüli) megjegyzés:

```
(setq tmode ;|valamilyen feljegyzés|; (getvar
"tilemode"))
```

A következő példa egy többsoros megjegyzést mutat be:

```
(setvar "orthomode" 1) ;|a megjegyzés itt kezdődik,
,és eddig a vonalig folytatódik,
de itt lent ér véget|; (princ "\nORTOMÓD bekapcsolva.")
```

AutoLISP program írása közben ajánlatos a megjegyzések használata. Erre jó példák találhatók *.lsp* kiterjesztésű, az AutoCAD programmal járó fájlban. Az *ai-utils.lsp* szép számban tartalmaz megjegyzéseket, illetve néhány érdekes és hasznos AutoLISP rutint is. A megjegyzésekkel kapcsolatban a következőkre érdemes odafigyelnie:

- A fejléc leíró része megfelelő részletességgel mutassa be a fájl használatát ahhoz, hogy az más dokumentációs anyagok felhasználása nélkül is érthető legyen.
- A megjegyzéseket megelőző pontosvesszők száma, valmint a megjegyzés elhelyezése utalhat a megjegyzés tartalmára, illetve annak fontosságára.

- Sok képernyő csak 80 karakter hosszúságú szövegeket tud megjeleníteni. Ezért előfordulhat, hogy a 81. oszloptól kezdődő megjegyzések nem jelennek meg a képernyőn.

Függetlenül attól, hogy milyen gyakorlatot alakított ki a megjegyzések használatára, nagyon fontos, hogy azok elhelyezése megfeleljen a mindenkori szövegkezelési szabályoknak.

Tabulálás és igazítás

A zárójelek túl gyakori használata megnehezítheti az AutoLISP program olvasását. Ennek elkerülésére a tabulálás a hagyományos megoldás. A tabulálás használatával úgy formázhatja a programsort, hogy annál nagyobb mértékben tolódjon jobbra az adott kódsor kezdete, minél mélyebben van beágyazva a programba. A beágyazás olyan mértékű is lehet, hogy a két szint közötti szabványos eltolódás lehet csupán néhány szóköznyi hosszú, ami jelentősen eltér a C nyelvben megszokott négy szóköznyi távolságtól.

Az AutoLISP a változók nevei, illetve a konstansok és a függvények nevei közötti esetleges több szóközt úgy kezeli, mintha csak egy lenne ott. A program számára a sor vége is szóközként jelenik meg.

A következő két kifejezés ugyanazt az eredményt adja.

```
(setq test1 123 test2 456)
(setq
    test1 123
    test2 456
)
```

Megjegyzés Az AutoLISP a tabulátorokat is szóközként értelmezi, mégsem ajánlott a használatuk. Egyes platformok eltérően értelmezhetik a tabulátorokat, ami kiszámíthatatlan eredményre vezethet.

A tabulálási szabályokat nehezebb megmagyarázni, mint példákon keresztül bemutatni. Az elrendezési szabályok elsajátításának legjobb módja, hogy elolvas és elemez olyan, mások által írt programokat, amelyeket könnyen tud olvasni. Emellett számos LISP szerkesztőrendszerben léteznek olyan "pretty printer" programok, amelyek képesek beolvasni és tabulálni az önkényesen szerkesztett LISP-kódokat.

AutoLISP változók

Egy AutoLISP változó a hozzárendelt érték adattípusát veszi fel. Amíg a változók új értéket nem kapnak, az eredeti értéküket tartják meg. Egy változónak az AutoLISP **setq** függvénnyel adhat értéket.

```
(setq változónév1 érték1 [változónév2 érték2 ...])
```

A **setq** függvény a meghatározott változónévhez rendeli a megadott értéket és a változó értékével tér vissza. A függvény visszatérési értékeként adja vissza a változó értékét. Amennyiben a **setq** függvényt a Parancs promptban használja, az AutoCAD beállítja a változó értékét, és kijelzi azt.

Parancs: **(setq val 3 abc 3.875)**
3.875
Parancs: **(setq layr "KÜLSŐ FALAK")**
"KÜLSŐ FALAK"
Parancs:

Minden változó a memóriaterület egy kis részét foglalja le, ezért érdemes programozáskor a változó nevét többször használni, vagy `nil`-re állítani, ha az értékére már nincs szükség. Amikor egy változó értékét `nil`-re állítja a lefoglalt memóriaterület felszabadul. Például, ha a val változóra nincs szüksége, akkor a lefoglalt memóriarészt a következőképpen szabadíthatja fel:

Parancs: **(setq val nil)**
nil

Változók használata a parancssorban

Amikor egy változónak értéket ad, a függvény visszatérési értékét felhasználhatja a parancssorban. Ez lehetővé teszi összetett nevek és számok tárolását és újrafelhasználását. Amikor az AutoCAD Parancs promptnál egy változó értékét akarja visszatérési értékként használni, adja meg a változó nevét és írjon elé egy felkiáltójelet (!). Például, ha egy olyan promptnak, amely valós értéket vár, az **abc** változó értékét kívánja megadni, a !abc parancsot írja be.

Oszloptávolság: **!abc**

Ha például a rajzon egy KÜLSŐ FALAK nevű fólia található, és ezt a FÓLIA paranccsal aktualizálni kívánja, elég, ha csak a változozó nevét adja meg a teljes fólianév helyett.

Parancs: fólia
?/Készít/beilleszt/Új/BE/KI/Szín/Vtípus/Fagyaszt/Olvaszt/Zár/Nyit: **s**
Új aktuális fólia <0>: **!fóli**
?/Készít/beilleszt/Új/BE/KI/Szín/Vtípus/Fagyaszt/Olvaszt/Zár/Nyit:
Parancs:

AutoCAD Parancs promptoknál értékként AutoLISP kifejezést is megadhat. Amennyiben egy értéket több promptnál is fel kíván használni írja a következőket:

Szög: **(setq ang 47.338)**

Ennek két következménye van: az **ang** változóhoz a 47,338 értéket rendeli, és azt a Szög promptnál visszatérési értékként adja meg. Ha legközelebb ezt az értéket akarja használni, elég beírnia, hogy **!ang**.

Mivel közönséges karakterláncok is kezdődhetnek ! vagy (karakterekkel, a TEXTEVAL változó értékét mindenképpen 1-re kell állítani, ha azt akarja, hogy egy változót vagy kifejezést a SZÖVEG vagy az ATTDEF parancsok értelmezzenek. Nem használhat változót vagy kifejezést a párbeszédpanelekben, és akkor sem, ha a DSZÖVEG parancsnak kíván karakterláncot továbbítani. Változót AutoCAD parancs kiadására sem használhat. Például, ha az **x** változó értéke a "vonal" karakterlánc, akkor a **!x** parancs megadása esetén az AutoCAD Parancs promptnál visszatérési értékként a "vonal" érték jelenik meg. A VONAL parancsot azonban nem hajtja végre a program. A command függvény AutoLISP kifejezésekből hajt végre AutoCAD parancsokat. További tudnivaló a "**command**" függvénynél, 13.fejezetben található.

Előre definiált változók

Az AutoCAD három, az AutoLISP alkalmazásokban felhasználható, előre definiált változót ad meg.

PAUSE	Olyan karakterláncként definiált változó, mely egyetlen visszahajló törtjelet (\) tartalmaz. A command függvénnyel használatos; hatására a program bemeneti adatot vár.
PI	A π (pi) konstans értékeként definiált változó. Közelítőleg 3,1415926.
T	Konstans T-ként definiált változó. Nem `nil` értékként használatos.

Megjegyzés Ezen változók értékét a **setq** függvénnyel megváltoztathatja. Mivel azonban más alkalmazások feltételezhetik, hogy értékük állandó, módosításuk nem ajánlott.

Számok kezelése

Az AutoLISP rendelkezik egész és valós számokat kezelő függvényekkel. Ezek a következők:

+ (összeadás)	abs	gcd	minusp
– (kivonás)	atan	log	rem
* (szorzás)	cos	logand	sin
/ (osztás)	exp	logior	sqrt
~ (bitenkénti NOT)	expt	lsh	zerop
1+ (növelés)	fix	max	

1– (csökkentés)	float	min

Az aritmetikai függvények leírását a 13. fejezetben találja.

Az AutoCAD aritmetikai függvényeivel összetett matematikai számításokat is végezhet alkalmazásaiban, a mindennapos használat során. Tegyük fel, hogy acél kötőelemek kapcsolódását kívánja részletesen ábrázolni. Ha egy 2,5" hosszú, 0,5" átmérőjű csavart rajzol, amelynek menetszáma hüvelyenként 13, és tudni szeretné, hogy hány menet van a csavaron, használja a * (szorzás) függvényt a Parancs promptnál.

Parancs: **(* 2.5 13)**
32.5

A *number* argumentummal rendelkező aritmetikai függvények (szemben például a *num* vagy *angle* típusúakkal) különböző értékeket adnak vissza attól függően, hogy egész vagy valós számot adunk meg argumentumként. Amennyiben egy függvény minden argumentuma egész szám, a végeredmény is egész szám marad. Azonban ha egy, vagy az összes argumentum valós, a végeredmény is valós szám lesz. Annak érdekében, hogy a végeredmény valós szám legyen, a műveletben legalább egy argumentumnak valósnak kell lennie.

Parancs: **(/ 12 5)**
2
Parancs: **(/ 12.0 5)**
2.4

Bár az AutoLISP 32 bites értékeket használ, az AutoCAD és az AutoLISP közötti kommunikációban csak 16 bites értékek vesznek részt. Ezért a +32767 -nél nagyobb, illetve a -32768 -nál kisebb értékeket az AutoLISP nem adja át az AutoCAD programnak. Amennyiben olyan értékre van szüksége, mely ezeket a határokat túllépi, alakítsa a számot a `float` függvénnyel hiszen valós számok 32 biten továbbítódnak.

Parancs: **(setq 32bit (float (* 3 30000)))**
90000.0

Karakterlánc-kezelés

Az AutoLISP rendelkezik karakterláncokat kezelő függvényekkel. Ezek a következők:

strcase	strlen	wcmatch
strcat	substr	

A következő függvények a karakterláncokat számokká és a számokat karakterláncokká konvertál. További tudnivaló a "Konverziók" című résznél, a 243. oldalon.

angtof	atof	distof
angtos	atoi	itoa
ascii	chr	rtos

A következő függvények karakterlánc-értékeket jelenítenek meg a parancssorban . Leírásukat a "Képernyővezérlés" című résznél, a 227. oldalon találja.

prin1	print
princ	prompt

A **strcase** függvény egy karakterlánc összes alfabetikus karakterét kis- vagy nagybetűssé konvertálja. Két argumentumot fogad el: egy karakterláncot, és egy opcionális argumentumot, amely megadja, hogy a konvertált szöveg kis- vagy nagybetűkből álljon. Amennyiben ez a választható argumentum hiányzik, akkor értéke nil, és a **strcase** függvény minden karaktert nagybetűvé konvertál.

Parancs: **(strcase "Ez egy PRÓBA.")**
"EZ EGY PRÓBA."

Amennyiben a második argumentum adott, és értéke nem `nil`, a karakterek kisbetűk lesznek. Az AutoLISP a `T` változót definiálja azokra az esetekre, ahol a program nem `nil` értéket használ igaz/hamis kapcsolóként. Ennek használata azonban nem kötelező, ha az alkalmazás számára megfelelőbb használhat más, nem `nil` értéket is.

Parancs: **(strcase "Ez egy PRÓBA." T)**
"ez egy próba."

A **strcat** függvény több karakterláncot egyetlen karakterlánccá alakít át. Ez akkor előnyös, ha változókarakterláncot kell konstans karakterláncba fűzni. A következő programrészlet egy karakterlánc-értékű változót hoz létre, majd a **strcat** függvény segítségével egy karakterlánc közepébe helyezi azt.

Parancs: **(setq str "NAGY") (setq bigstr (strcat "Ez egy " str " próba."))**
"Ez egy NAGY próba."

Egy karakterlánc maximum 132 karakter hosszú lehet. A **strcat** függvény több karakterlánc összefűzésével korlátlan hosszúságú karakterlánc létrehozására képes.

Amennyiben a `bigstr` változó értéke az előző karakterlánchoz van beállítva, a **`strlen`** függvény megadja annak karakterszámát (a szóközöket is beleértve).

Parancs: **(strlen bigstr)**
19

A **`substr`** függvény a karakterlánc egy darabját adja meg. Két kötelező és egy választható argumentuma van. Az első karakterlánc. A második egy egész szám, ami megadja a karakterlánc első karakterét, azt amelyikkel a levágott karakterláncnak kell kezdődnie. Amennyiben a harmadik argumentum is adott, az a levágott karakterlánc karaktereinek számát adja meg. Ha nincs megadva, akkor a levágott karakterlánc a megadott kezdőkarakter után következő, összes karaktert tartalmazza.

A **`substr`** függvény lehetőséget ad a fájl nevének és hárombetűs kiterjesztésének szétválasztására is. Először rendelje a változót egy fájlnévhez. (Az alkalmazásától függően ezt általában úgy teheti meg, hogy lekérdez egy rendszerváltozót, vagy felhasználói bevitelt kér.)

Parancs: **(setq filnam "nagyfájl.txt")**
"nagyfájl.txt"

Tegyük fel, hogy olyan karakterláncra van szüsége, amely az utolsó négy karakter (a pont és a három betűből álló kiterjesztés) kivételével az összes karaktert tartalmazza. A **`strlen`** függvény megadja a karakterlánc hosszát, amiből le kell vonnia négyet. Ezután a **`substr`** függvénynek adja meg a levágott karakterlánc első karakterét és hosszát.

Parancs: **(setq newlen (- (strlen filnam) 4))**
7
Parancs: **(substr filnam 1 newlen)**
"nagyfájl"

Amennyiben az alkalmazásban nincsen szükség a `newlen` változó értékére, az előbbi két programsort egy sorban is meg tudja adni.

Parancs: **(substr filnam 1 (- (strlen filnam) 4))**
"nagyfájl"

Vezérlőkarakterek a karakterláncokban

Idézőjeles karakterláncokon belül a visszahajló törtjel (\) lehetővé teszi vezérlõkarakterek (escape kódok) használatát. A következő táblázat tartalmazza a jelenleg használatos vezérlőkaraktereket.

Vezérlőkarakterek

Kód	*Leírás*
\\	\ karakter
\"	" karakter
\e	Escape karakter
\n	Újsor karakter
\r	Return karakter
\t	Tab karakter
nnn	*nnn* oktális kódú karakter
\U+*xxxx*	Unicode karaktersorozat
\M+*nxxxx*	Multibájt karaktersorozat

A **prompt** függvény kiterjeszti a vezérlőkaraktereket, és kijelzi őket a parancssorban. A **prompt** függvény egyetlen argumentumot,egy karakterláncot fogad el. Ezt a karakterláncot a parancssorban jeleníti meg, miközben visszatérési értéke nil. A karakterlánc minden vezérlőkaraktere kibővítésre kerül. További tudnivaló "**prompt**" függvény, 13. fejezet.

Amennyiben egy idézőjeles karakterláncon belül a visszahajló törtjel (\) vagy az idézőjel (") karaktert kívánja használni, mindig tegyen eléjük egy visszahajló törtjelet (\).

Parancs: **(prompt "A \"fájlnév\" : D:\\ACAD\\TEST.TXT. ")**
A "fájlnév" : D:\ACAD\TEST.TXT. nil

A (\e) vezérlőkarakter használható arra is, hogy escape sorozattal utasítsa az operációs rendszert a képernyő grafikájának kicserélésére, a vezérlőbillentyűk mozgatására vagy a billentyűzet újradefiniálására. Az operációs rendszer leírásában megtalálja a rendszer által felismert escape sorozatot. DOS rendszerekben az escape sorozatokat gyakran a billentyűzet újradefiniálására használják. A következő példában a **prompt** függvény a **qsave** ↵ karakterláncot az F12 billentyűhöz rendeli.

Parancs: **(prompt "\e[0;134;'qsave';13p")**
nil

Az előző példában a karakterlánc négy részből áll. Az **\e[** jelzi a DOS rendszernek, hogy escape sorozat következik, a **0;134;** kód pedig az F12 billentyűt jelöli ki. Amikor az F12 billentyűt lenyomja a

karakterlánc **'qsave';** része a Parancs prompthoz kerül, a **13p** pedig azt adja meg, hogy a karakterláncot ↵ követi. Amennyiben a Parancs promptnál szereplő karakterlánc idézőjelben van és pontosvessző követi, a **qsave** helyett bármely érvényes karakterlánc használható.

Ha egy adott helyen meg kívánja törni a karakterláncot, használja az újsor (`\n`) karaktert.

```
Parancs: (prompt "Egy példa az \nújsor karakterre. ")
Egy példa az

újsor karakterre. nil
```

A visszatérés karakter (`\r`) az aktuális sor kezdetéhez tér vissza. Ez olyan bővülő információk kijelzésekor hasznos, mint például a használt objektumok száma.

Amikor az AutoLISP kiértékel egy kifejezést, az utolsó kifejezés értékét adja vissza. Mivel a **`prompt`** függvény visszatérési értéke nil, a fenti példák visszatérési értékként minden esetben a nil értéket írják ki a karakterlánc után. A **`princ`** függvény is karakterláncot ír ki a parancssorba, de visszatérési értéke nem nil, hanem maga a kifejezés (ebben az esetben karakterlánc). A **`princ`** függvény abban is különbözik a **`prompt`** függvénytől, hogy van választható argumentuma is, ami karakterláncon kívül más adattípus is lehet. Ezért ha az éppen használt AutoLISP kifejezés olyan **`princ`** függvény hívásával végződik, aminek egyáltalán nincsen argumentuma, a visszatérési `nil` érték nem jelenik meg (mivel nincs, mit visszaadni). Ezt csendes kilépésnek nevezik.

A tabulátor karakter (\t) karakterláncok tabulására, vagy más, tabulátor karaktereket tartalmazó szövegnek az adott karakterlánchoz történő igazítására használható. A következő példában figyelje meg, hogy a **`princ`** függvény nem ad ki `nil` értéket.

```
Parancs: (prompt "\nNév\tIroda\n------\t------
1> \nSue\t101\nJoe\t102\nSam\t103\n") (princ)
Név     Iroda
----------
Sue     101
Joe     102
Sam     103
```

A *\nnn* vezérlőkaraktereket használhatja olyan karakterek megjelenítésére, amelyek nem szerepelnek a szabvány billentyűzeten. Az *nnn* a megjeleníteni kívánt karakter oktális kódja. Az A függelékben megtalálja a szabvány ASCII karakterek felsorolását (ezek decimális kódja 0-tól és 127-ig terjed), a karakterek decimális, oktális és hexadecimális kódjait. A kiterjesztett 128 és 256 közötti decimális kódú karakterekhez tartozó kódok az installált karakterbeállítástól függenek. A kiterjesztett karakterekhez tartozó kódok platformonként

változnak, és változhatnak egyazon platformon belül különböző alkalmazásokban is. Az "ASCII kódok konverziója" című résznél, a 246. oldalon talál egy minta-alkalmazást, amely AutoCAD szövegablakban megjeleníti az összes, az Ön rendszerében fellelhető karaktert, majd egy fájlba írja őket.

Windows

Amennyiben a grafikus ablak a Courier fontot használja, a következő kód jeleníti meg a " ± " jelet.

Parancs: **(prompt "Az érték: \261 7001. ") (princ)**
Az érték: ± 7001.

DOS

A következő kód jeleníti meg a " ± " jelet.

Parancs: **(prompt "Az érték: \361 7001. ") (princ)**
Az érték: ± 7001.

Összehasonlítás globális karakterekkel

Az alkalmazások a **wcmatch** függvény segítségével összehasonlíthatnak egy karakterláncot egy globális karaktereket tartalmazó mintával. Ez a lehetőség használható egy kiválasztási halmaz összeállításakor (az **ssget** függvény használatakor), valamint bővített elemadatok alkalmazásnév szerinti lekérdezésekor (az **entget** függvény használatakor).

A **wcmatch** függvény egyetlen karakterláncot hasonlít össze egy mintával. Ha a karakterlánc egyezik a mintával, akkor `T`, egyébként `nil` értékkel tér vissza. A globális karakterekből álló minták hasonlóak azokhoz a *szabályos kifejezésekhez*, amelyeket sok operációs rendszer és alkalmazás használ. A mintákban az alfabetikus karakterek és a számok betű szerint kerülnek feldolgozásra. Szögletes zárójelek használhatók opcionális karaktereknek vagy betűkből és számjegyekből álló tartománynak a megadására. A kérdőjel (?) egyetlen karakternek, a csillag (*) egy karaktersorozatnak felel meg. Néhány egyéb különleges karakternek szintén speciális jelentése van a mintán belül. Ha a keresési minta elején és végén a * karaktert használja, a kívánt részt bárhol megtalálhatja, bárhol is van a karakterláncon belül. További részletek találhatók a **wcmatch** függvény ismertetésénél, a 13. fejezetben.

Az alábbi példában a hasonl nevű változó értékét állítjuk be:

Parancs: **(setq** hasonl **"ez egy karakterlánc - próba1 próba2 vége")**
"ez egy karakterlánc - próba1 próba2 vége"

A következő függvényhívás azt vizsgálja, hogy a hasonl változó az "ez" karakterekkel *kezdődik-e:*

Parancs: **(wcmatch hasonl "ez*")**
T

A következő függvényhívás a szögletes zárójelek használatát szemlélteti a mintában. Az alábbi esetben a **wcmatch** függvény T értékkel tér vissza, amennyiben a hasonl változó tartalmazza a "próba4", "próba5", "próba6" vagy "próba9" valamelyikét (figyelje meg a "*" jel használatát)

Parancs: **(wcmatch hasonl "*próbatest[4-69]*")**
nil

Ezzel szemben:

Parancs: **(wcmatch hasonl "*próba[4-61]*")**
T

mivel a karakterláncban előfordul a "próba1".

Vesszőkkel elválasztva több minta is megadható egyszerre. A következő függvényhívás akkor tér vissza T értékkel, ha a hasonl változó egyenlő az "ABC" karakterlánccal, vagy ha az "XYZ" karakterekkel *kezdődik*, illetve a "vége" karakterekre *végződik*.

Parancs: **(wcmatch hasonl "ABC,XYZ*,*vége")**
T

Relációs függvények

Az AutoLISP tartalmaz olyan függvényeket, melyek ellenőrzik az egyenlőséget, illetve lehetővé teszik a relációs műveletek csoportosítását. A relációs függvények a következők:

= (egyenlő)	and	or
/= (nem egyenlő)	Boole	repeat
< (kisebb mint)	cond	while
<= (kisebb vagy egyenlő)	eq	
> (nagyobb mint)	equal	
>= (nagyobb vagy egyenlő)	if	

Az összes relációs függvény leírását megtalálja a 13.fejezet "AutoLISP függvények katalógusa" című fejezetben.

Listakezelés

Az AutoLISP tartalmaz listák kezelésére alkalmas függvényeket is. A listakezelő függvények a következők.

append	foreach	listp	reverse
assoc	last	mapcar	subst
car and cdr	length	member	
cons	list	nth	

Ez a rész az **append**, **assoc**, **car**, **cons**, **list**, **nth** és **subst** függvények használatára mutat be példákat. A listakezelő függvények leírását megtalálja a 13. "AutoLisp függvények katalógusa" című fejezetben.

A listák által számos, egymással összefüggő értéket tárolhat hatékonyan. Több AutoLISP függvény biztosít lehetőséget két- illetve háromdimenziós grafikai alkalmazások programozására. Ezek a függvények a pontok értékeit listák formájában adják vissza.

A **list** függvénnyel egyszerűen lehet összetartozó adatokat csoportosítani. Nem kell hogy ezek az adatok hasonló típusúak legyenek. A következő programsor három, összetartozó adatot csoportosít egy listába.

Parancs: **(setq lst1 (list 1.0 "Egy" 1))**
(1.0 "Egy" 1)

A listából az **nth** függvény segítségével vehet ki adatot. Az **nth** függvénynek két argumentuma van. Az első egy egész szám, ami azt az adatot határozza meg, amit a listából ki akar venni. A 0 adja meg a lista első adatát, az 1 a másodikat, és így tovább. A második argumentum maga a lista. A következő kód az `lst1` lista második adatával tér vissza.

Parancs: **(nth 1 lst1)**
"Egy"

A **car** és a **cdr** függvények a fentiektől eltérő lehetőséget nyújtanak adatok listákból történő kiemelésére. Ezen függvények használatára a következő, "Pontlisták" című részben talál példákat.

Három függvény ad lehetőséget már létező listák módosítására. Az **append** függvény úgy adja vissza a listát, hogy új adatokat tesz a lista végére, a **cons** függvény pedig úgy, hogy a lista elejére teszi az új adatokat. A **subst** függvény olyan listát ad vissza, melyben egy meghatározott régi adatot, minden előfordulási helyén egy új adat helyettesít. Ezek a függvények nem módosítják az eredeti listát, hanem egy új, módosított listát hoznak létre. Ahhoz, hogy az eredeti listát módosítsa, azt helyettesítenie kell a módosított listával.

Az **append** függvény bármely számú különböző lista egyetlen listaként való kezelésére alkalmas. Ezért a függvény minden argumentumának listának kell lennie. Az alábbi program egy újabb "Egy" adatot fűz az 1st1 listához. Emlékeztetőül, a quote (vagy ') függvény egyszerű megoldást nyújt az "Egy" karakterlánc listába történő befűzésére.

Parancs: **(setq lst2 (append lst1 '("Egy")))**
(1.0 "Egy" 1 "Egy")

A **cons** függvény egyedülálló elemek és listák összekapcsolására szolgál; újabb "Egy" karakterláncot tesz az új, 1st2 nevű lista elejére.

Parancs: **(setq lst3 (cons "Egy" lst2))**
("Egy" 1.0 "Egy" 1 "Egy")

A **substr** függvény egy listán belül egy meghatározott adatot, minden egyes előfordulási helyén új adattal helyettesít. A következő program minden "Egy" karakterláncot az "egy" karakterlánccal helyettesít.

Parancs: **(setq lst4 (subst "Egy" "egy" lst3))**
("egy" 1.0 "egy" 1 "egy")

Pontlisták

Az AutoLISP a következő előírások betartását figyeli grafikus koordináták kezelése esetén. A pontok két- vagy háromtagú zárójeles *listák*ként jelennek meg.

Kétdimenziós pontok Két valós számból álló listák (sorrendben X és Y) például

```
(3.4 7.52)
```

háromdimenziós pontok Három valós számból álló listák (sorrendben X, Y és Z) például

```
(3.4 7.52 1.0)
```

A pontváltozók az X, Y és (tetszőlegesen) Z elemeket tartalmaznak. A lista első eleme a pont X koordinátája, második eleme a pont Y koordinátája és harmadik eleme (ha van) a pont Z koordinátája. A **list** függvénnyel készíthet ilyen listákat.

Parancs: **(list 3.875 1.23)**
(3.875 1.23)
Parancs: **(list 88.0 14.77 3.14)**
(88.0 14.77 3.14)

Ha megadott koordinátákat kíván egy pontváltozóhoz rendelni, használja a következő kifejezések valamelyikét:

Parancs: **(setq pt1 (list 3.875 1.23)**
1> **pt2 (list 88.0 14.77 3.14)**
1> **abc 3.45**
1> **pt3 (list abc 1.23))**
(3.45 1.23)

Az utóbbi kifejezés az `abc` változó értékét használja a pont X koordinátájaként.

Amennyiben a lista minden eleme konstans érték, a **`list`** függvény helyett a **`quote`** függvényt is használhatja a lista definiálására. A **`quote`** függvény kiértékelés nélkül adja vissza a kifejezést.

Parancs: **(setq pt1 (quote (4.5 7.5)))**
(4.7 7.5)

A **quote** függvényt egyszeri aposztróffal (**'**) helyettesítheti. A következő program ugyanazt eredményezi, mint a fenti.

Parancs: **(setq pt1 '(4.5 7.5))**
(4.7 7.5)

Egy pont X, Y, és Z koordinátákra külön-külön is hivatkozhat a **`car`**, a **`cadr`** és a **`caddr`** beépített függvényekkel. A következő példa azt mutatja meg, hogy egy háromdimenziós pontlistából hogyan veheti ki az X, Y, és Z koordinátákat. A `pt` változó egy pontváltozó, a következő értékekkel: (1.5 3.2 2.0).

Parancs: **(setq pt '(1.5 3.2 2.0))**
(1.5 3.2 2.0)

A **`car`** függvény a lista első tagját adja meg. Ebben a példában a függvény az `x_val` változónak adja át a `pt` pont X koordinátáját.

Parancs: **(setq x_val (car pt))**
1.5

A **`cadr`** függvény a lista második tagját adja meg. Ebben a példában a függvény az `y_val` változónak adja át a `pt` pont Y koordinátáját.

Parancs: **(setq y_val (cadr pt))**
3.2

A **`caddr`** függvény a lista harmadik tagját adja meg. Ebben a példában a függvény az `z_val` változónak adja át a `pt` pont Z koordinátáját.

Parancs: **(setq z_val (caddr pt))**
2.0

A következő program egy négyszög bal alsó és jobb felső (`pt1` és `pt2`) sarkait adja meg.

Parancs: **(setq pt1 '(1.0 2.0) pt2 ' (3.0 4.0))**
(3.0 4.0)

A **car** és a **cadr** függvényekkel a `pt3` változó értékének a négyszög bal felső sarkának koordinátáit adhatja meg úgy, hogy kivonja a `pt1` változó X koordinátáját és a `pt2` változó Y koordinátáját.

Parancs: **(setq pt3 (list (car pt1) (cadr pt2)))**
(1.0 4.0)

Az előző kifejezés a `pt3` változó értékének a (1,0 4,0) pontot adja meg.

Kapcsolt értékpárok

Az AutoCAD más típusú adatlisták összeállítására is képes. Ezeket a speciális listákat *kapcsolt értékpároknak* nevezik. A kapcsolt értékpár olyan lista, amelynek mindig két tagja van. Az ilyen listák megjelenítésekor az AutoLISP mindig pontot, vagy vesszőt tesz a lista két tagja közé. A listakezelő függvények nagy része nem fogadja el a kapcsolt értékpárokat, ezért mindig győződjön meg arról, hogy a megfelelő típusú listát adta át a függvényeknek.

A **cons** fügvény azon kívül, hogy egy lista elejéhez illleszt egy tételt, képes kapcsolt értékpárok létrehozására is. Amennyiben a **cons** második argumentuma egy atom (konstans vagy idézőjeles kifejezés), a függvény kapcsolt értékpárt hoz létre.

Parancs: **(setq d1 (cons 1 "Egy"))**
(1 . "Egy")

A kapcsolt értékpárokat a **car**, a **cdr**, és az **assoc** fügvények segítségével kezelheti. A következő kód egy asszociációs listát hoz létre. Az asszociációs lista további listákat tartalmaz. Az AutoCAD így tárolja a rajzelemek definíciós adatait (a rajzelem-definíciós adatokról bővebben a 9., "Kiválasztási halmazokat, objektumokat és szimbólumtáblákat kezelő függvények" című fejezetben talál leírást). A következő példa egy kapcsolt értékpárokból álló asszociációs listát hoz létre.

Parancs: **(setq d2 (list d1 (cons 2 "Kettő")(cons 3 "Három")))**
((1 . "Egy") (2 . "Kettő") (3 . "Három"))

Az **assoc** függvény az asszociációs listának egy meghatározott listáját adja meg, függetlenül attól, hogy az adott lista az asszociációs listán belül hol helyezkedik el. Az **assoc** függvény a listában egy meghatározott kulcselemet keres.

Parancs: **(assoc 2 d2)**
(2 . "Kettő")

Szimbólum- és függvénykezelés

Az AutoLISP rendelkezik a szimbólumokat és változókat kezelő függvényekkel. A szimbólumkezelő függvények a következők:

atom	not	quote	type
atoms-family	null	set	
boundp	numberp	setq	

Minden szimbólumkezelő függvény leírását megtalálja a 13. , az "AutoLISP függvények katalógusa" című fejezetben.

Az AutoLISP lehetőséget nyújt egy vagy több függvénycsoport kezelésére. A függvénykezelő függvények a következők:

apply	eval	progn	untrace
defun	lambda	trace	

Ez a rész a **defun** függvény alkalmazására mutat példákat. Minden függvénykezelő függvény leírását megtalálja a 13., az "AutoLISP függvények katalógusa" című fejezetben.

Az AutoLISP lehetőséget nyújt Önnek saját függvények definiálására. Ezeket a felhasználó által definiált függvényeket definiálásuk után ugyanúgy használhatja, a parancssorban vagy AutoLISP kifejezésekben, mint az AutoLISP saját függvényeit. Ugyanígy létrehozhat AutoCAD parancsokat is, mivel a parancsok tulajdonképpen egyfajta speciális függvények.

A **defun** függvény (részletesebb leírása a 13. fejezetben) több kifejezést kapcsol össze függvénnyé vagy paranccsá. Ennek a függvénynek legalább három argumentuma van, melyek közül az első a függvény nevét adja meg. A második argumentum egy argumentumlista (a függvény által használt argumentumok és lokális szimbólumok listája). Az argumentumlista lehet `nil` értékű vagy üres lista ( ) is. Az argumetumlisták részletes leírását az "Argumentummal rendelkező függvények" című résznél, a 215. oldalon találja. A lokális változókat az argumentumoktól egy törtjel (/) választja el. A lokális szimbólumok részletes leírását a "Lokális szimbólumok a függvényekben" című résznél, a 214. oldalon találja. Az első két argumentumot a függvényt alkotó kifejezések követik. Egy függvénydefinícióban legalább egy kifejezésnek szerepelnie kell.

```
(defun szimbólumnév ( argumentumok / lokális_szimbólumok )
  kifejezések
)
```

A következő kód egy egyszerű argumentum nélküli függvényt definiál, amely egy üzenetet jelenít meg a parancssorban. Figyelje meg, hogy az argumentumlista üres (()).

Parancs: **(defun DONE ( ) (prompt "\nviszlát! "))**
DONE

Miután definiálta a `DONE` függvényt, ugyanúgy használhatja, mint bármelyik más függvényt. Mivel ez a függvény a viszlát! üzenetet egy új sorba írja ki, a következőképpen is használhatja:

Parancs: **(prompt "Az érték: 127. ")(DONE)(princ)**
Az érték: 127
viszlát!

Az argumentum nélküli függvények feleslegesnek tűnhetnek. Arra azonban megfelelnek, hogy bizonyos rendszerváltozók vagy feltételek állapotát lekérdezzék, és az erre utaló értékkel térjenek vissza.

Megjegyzés A nem kapcsolódó függvényekkel kapcsolatos problémák elkerülése érdekében használhatja az `S::` előtagot. Ez annyit jelent, hogy az adott név "foglalt". Ezt az előtagot csak a S::STARTUP függvény esetében használja. A függvény leírását a 206. oldalon, az "S::STARTUP függvény -automatikus végrehajtás" című részben találja.

Az AutoCAD minden új szerkesztés kezdetekor automatikusan betölti az Ön által definiált függvényeket. (További tudnivaló az "Automatikus betöltés és végrehajtás" című résznél, a 162. oldalon található).

C:*XXX* függvények

Ha a függvény a `C:XXX` formátumú névvel van definiálva, akkor az a beépített AutoCAD parancsokhoz hasonlóan a Parancs promptnál adható ki. Ezt a tulajdonságot új parancsok létrehozásánál, és a már meglévők átdefiniálásakor használhatja.

Ha a függvényeket AutoCAD parancsként kívánja használni a következő szabályokat tartsa be:

- A függvény neve csak `C:XXX` alakú lehet (nagy- vagy kisbetűkkel). A név `C:` részének mindig jelen kell lennie; az `XXX` rész egy Ön által választott parancsnév. A `C:XXX` függvényekkel a beépített AutoCAD parancsok felülírhatók. (További tudnivaló az "AutoCAD parancsok átdefiniálása" című résznél, a 213. oldalon)

Megjegyzés Ebben az esetben a C: nem egy lemezmeghajtóra való hivatkozás, hanem egy speciális előtag, amely egy parancsként végrehajtható függvényt jelöl.

- A függvényeket argumentumok nélkül kell definiálni. (A lokális szimbólumok megengedettek.)

Az így definiált függvény bármely beépített AutoCAD parancs bármelyik promptjánál transzparens módon meghívható, feltéve, hogy a függvényben nem szerepel a **command** függvényhívás. Egy **C:XXX** módon definiált parancs transzparens kiadásához az **XXX** név elé a (') (aposztróf) jelet kell tennie.

Beépített parancsot úgy adhat ki transzparens módon, mialatt **C:XXX** parancs aktív, hogy egy aposztróf (') jelet tesz elé - ugyanúgy, mint az összes transzparens módon kiadott parancs esetében. A **C:XXX** parancsot viszont nem adhatja ki transzparens módban, mialatt a **C:XXX** parancs aktív. Egy olyan függvény meghívásakor, amely egy másik AutoLISP függvény programjából, parancsként van definiálva, az egész nevet kell használnia - például, **(C:HELLO)**.

Ha a függvény definícióját a **setfunhelp** függvény meghívása követi, egy felhasználó által definiált paranccsal hozzárendelhet egy Súgófájlt és egy témát. Ha a felhasználó a promptnál, a felhasználó által definiált paranccsal, transzparens módban hívja meg a súgót ('**súgó**') a **setfunhelp** által meghatározott téma jelenik meg.

Az AutoLISP által definiált parancs promptjaira általában nem válaszolhat másik AutoLISP kifejezéssel. Amennyiben azonban egy AutoLISP rutinban az **initget** függvényt alkalmazza, akkor bizonyos függvények esetében tetszőleges billentyűzetbevitelt engedhet meg. Ez az AutoLISP alapú parancsok számára is lehetővé teszi, hogy AutoLISP kifejezéseket fogadjanak el válaszként. Ezenkívül a DIESEL kifejezések által megadott értékek is képesek arra, hogy bizonyos mértékben kiértékeljék az aktuális rajzot, és az AutoLISP progrmnak továbbítsák az értékeket. További információkat talál a DIESEL Karakteres kifejezésnyelvről az 5., "Státuszsor konfiguráció és a DIESEL karakteres kifejezésnyelv" című fejezetben.

Parancsok készítése

A **C:XXX** típusú függvényekkel definiálhatók egyszerű üzenetet megjelenítő parancsok is.

Parancs: **(defun C:HELLO () (prompt "Helló világ. \n") (princ))**
C:HELLO

A HELLO most parancsként van definiálva.

Parancs: **hello**
Helló világ.

Az új parancs transzparens módon is kiadható, mivel nem hívja meg a **command** függvényt.

Parancs: **vonal**
Ponttól: **'hello**
Helló világ.
Ponttól:

AutoCAD parancsok átdefiniálása

Az AutoLISP, a külső parancsok és az álnév jellemző segítségével saját AutoCAD parancsokat definiálhat. A DEFTÖRÖL paranccsal beépített AutoCAD parancsokat definiálhat újra úgy, hogy a saját maga által definiált parancsoknak ugyanazt a nevet adja. A beépített definició visszahívására a RÉGIDEF parancsot használja. A DEFTÖRÖL parancs érvényessége mindig csak az adott szerkesztési cikusban tart.

Újradefiniált parancsot bármikor aktivizálhat, csak meg kell adnia az igazi nevet úgy, hogy a parancs neve elé egy pontot tesz. Például, ha a KILÉP függvényt újradefiniálta, az eredeti parancsot a **.kilép** kiadásával hajthatja végre.

Íme egy példa. Azt szeretné, hogy a VONAL parancs kiadásakor az AutoCAD mindig figyelmeztesse a VLÁNC parancs kiadására. Az eredeti VONAL parancsot a következőképpen helyettesítheti a saját maga által definiált C:VONAL függvénnyel:

Parancs: **(defun C:VONAL ( )**
1> **(princ "Nem kellene használnia a VLÁNC parancsot?\n")**
1>(command ".VONAL") **(princ))**
C:VONAL

Ebben a példában a C:LINE függvény először kiírja a megadott üzenetet, majd végrehajtja az eredeti VONAL parancsot (az igazi nevét használva: .VONAL). Mielőtt az AutoCAD használná a VONAL parancs új definícióját, mindenképpen helyettesítenie kell a beépített VONAL parancsot. (Azt, hogy az AutoCAD milyen sorrendben ismeri fel a parancsneveket a "Parancskeresési eljárás" című résznél, a 22. oldalon találja meg.)

Parancs: **deftöröl**
Parancsnév: **vonal**

Mostantól ha a Parancs promptba beírja: **vonal**, az AutoCAD a C:LINE AutoLISP függvényt használja.

Parancs: vonal
Nem kellene használnia a VLÁNC parancsot ?
.VONAL ponttól:
Ponttól:

Az előző példa feltételezi, hogy a CMDECHO rendszerváltozó értéke 1, azaz be van kapcsolva. Amennyiben a CMDECHO értéke 0, azaz ki van kapcsolva, az AutoCAD nem jeleníti meg azokat a promtokat, amelyek a **command** függvény hívásának eredményei. A következő kód a CMDECHO rendszerváltozót használja a parancssorok kijelzésének vezérlésére.

```
Parancs:  (defun C:VONAL ( )
1> (setvar "cmdecho" 0)
1> (princ "Nem kellene használnia a  VLÁNC parancsot?\n")
1> (command ".VONAL") (setvar "cmdecho" 1) (princ) )
C:VONAL
```

Ezt az eljárást például egy rajzolórendszerben használhatja. Újradefiniálhatja az ÚJ, MEGNYIT, KILÉP és VÉGE parancsokat, hogy az azokhoz tartozó információkat naplófájlba írja még a szerkesztési ciklus vége előtt.

Megjegyzés A parancsokba írt pont előtag (.xxxx) használata ajánlott, hogy a menük, forgatókönyvek és AutoLISP programok védelme érdekében. Ez biztosítja, hogy a használt alkalmazás újradefiniált parancsok helyett beépített parancsokat használjon.

Lokális szimbólumok a függvényekben

Az AutoLISP lehetővé teszi, hogy definiálja azon szimbólumok (változók) listáját, amelyeket csak az Ön függvényei használnak. Ezeket hívják *lokális szimbólumok*nak. Ezeknek a szimbólumoknak a használata biztosítja azt, hogy a környező alkalmazások a függvény változóit ne befolyásolják, és hogy a változókat a függvény végrehajtása után ne lehessen használni.

Számos felhasználó által definiált függvény nagyobb alkalmazások segédfüggvénye. Emellett a felhasználó által definiált függvények általában sok olyan változót tartalmaznak, amelyek az értékei és használata csak az adott függvényre jellemző.

A következő példa bemutatja, hogyan használhatók lokális szimbólumok egy felhasználó által definiált függvényben. (A törtjel és a lokális szimbólum között legalább egy szóköznek kell lennie.)

```
Parancs:  (defun LOCAL ( / aaa bbb)
1> (setq aaa "A" bbb "B")
1> (princ (strcat "\naaa értéke  " aaa ))
1> (princ (strcat "\nbbb értéke  " bbb))
1> (princ) )
LOCAL
```

Mielőtt az új függvényt kipróbálná, adjon az aaa és bbb változóknak a **LOCAL** függvényben megadottól eltérő értékeket.

Parancs: **(setq aaa 1 bbb 2)**
2

A felkiáltójelet (!) használva meggyőződhet az `aaa` és `bbb` változók megfelelő beállításáról.

Parancs: **!aaa**
1
Parancs: **!bbb**
2

Most próbálja ki a **LOCAL** függvényt.

Parancs: **(local)**
aaa értéke A
bbb értéke B

Megfigyelheti, hogy a függvény az `aaa` és `bbb` változók lokális értékeit használja. A felkiáltójelet (!) használva meggyőződhet arról is, hogy az `aaa` és `bbb` változók az aktuális értékei nem a lokális értékek.

Parancs: **!aaa**
1
Parancs: **!bbb**
2

Ez a módszer egyrészt azt biztosítja, hogy a változók egy bizonyos függvény lokális értékei; másrészt, hogy az általuk használt memóriaterületet más függvények is használhatják.

Argumentummal rendelkező függvények

Az AutoLISP nyelvben argumentummal rendelkező függvények is definiálhatók. Szemben az eredeti AutoLISP függvénnyekkel, a felhasználó által definiált függvényeknek nem lehet választható argumentumuk. Amikor olyan, felhasználó által definiált függvényt használ, amelyiknek van argumentuma, minden argumentumnak értéket kell adnia.

Az argumentumként szolgáló szimbólumokat a lokális szimbólumok előtt, az argumentumlistában kell definiálni. Az argumentumok tulajdonképpen speciális lokális szimbólumok; az argumentum változók csak az adott függvényben használhatók. Egy függvényben nem definiálhat több, azonos nevű argumentumot.

A következő példa egy olyan függvényt definiál, amelynek két karakterlánc-argumentuma van. Ezeket egy harmadikkal illeszti össze, és eredményül az új karakterláncot adja vissza.

Parancs: **(defun ARGTEST (arg1 arg2 / ccc)**
1> **(setq ccc "Konstans karakterlánc")**
1> **(strcat ccc ", " arg1 ", " arg2))**
ARGTEST

Ezt a típusú függvényt egy adott alkalmazásban számtalanszor használhatja arra fel, hogy megadott sorrendben illesszen egymáshoz két karakterlánc-változót és egy konstans karakterláncot. Mivel a függvénynek van visszatérési értéke, ezt a későbbi felhasználás érdekében változóként mentheti el.

Parancs: **(setq newstr (ARGTEST "Karakterlánc 1" "Karakterlánc 2"))**
"konstans karakterlánc, Karakterlánc 1, Karakterlánc 2"

A newstr nevű értéke most változó a három, egymáshoz illesztett karakterlánc.

Hibakezelés

Az AutoLISP hibakezelő függvényekkel is rendelkezik. Ezek a következők:

alert *error* exit quit

Az összes hibakezelő függvény leírását megtalálja a 13. , az "AutoLISP függvények katalógusa" című fejezetben.

Az AutoLISP lehetőséget nyújt a felhasználói hibák (vagy program-hibák) kezelésére. Az **`*error*`** függvény segítségével gondoskodhat arról, hogy egy váratlan hiba fellépése után helyreálljon bizonyos rendszerváltozók beállítása, illetve az AutoCAD egy adott állapota. Ennek a felhasználó által definiált függvénynek a segítségével megvizsgálhatja a hiba okát, és megfelelő üzenetet küldhet a felhasználónak.

Amennyiben az AutoLISP hibát talál a kiértékelés alatt, a következő üzenetet írja ki:

Hiba: szöveg

Ebben az üzenetben a *szöveg* leírja a hibát. Ha az **`*error*`** függvény definiálva van (nem `nil`), az AutoLISP végrehajtja ezt a függvényt, (a *szöveg* egy egyszerű argumentumként adódik át). Amennyiben ez a függvény nincs definiálva, vagy nil, akkor az AutoLISP kiértékelése megszakad , és megjelenik a hívott függvény, valamint annak hívói

maximum 100 szint mélységben. A hibakeresés alatt érdemes ezt a hibakezelőt működtetni.

Az utolsó hiba kódja az AutoCAD ERRNO nevű rendszerváltozójában kerül elmentésre, ahonnan a **getvar** függvény segítségével kaphatja vissza. A hibaüzeneteket és a hibakódokat a 17., az "AutoLISP hibaüzenetek és hibakódok" című fejezetben találja. További tudnivalók az **"*error*"** címszót a 13. fejezetben.

A saját ***error*** függvény definiálása előtt mentse el az ***error*** aktuális tartalmát, így a programból történő kilépéskor visszaállíthatja az előző hibakezelőt. Amikor hibát észlel, az AutoCAD meghívja az aktuálisan definiált ***error*** függvényt, és egyetlen argumentumként egy karakterláncot ad át neki, amely a hiba természetére utaló szöveget tartalmazza. A saját ***error*** függvényt tipikusan úgy szokás kialakítani, hogy az Esc (mégsem) lenyomásakor vagy egy **exit** függvényhívás után "csendesen" lépjen ki. Ezt szabványos módon a hibakezelő rutinban elhelyezett alábbi utasításokkal lehet megvalósítani:

```
(if
  (or
    (/= msg "Függvény érvénytelenítve")
    (= msg "kilép megszakítás")
  )
  (princ)
  (princ (strcat "\nHiba: " msg))
)
```

Ez a programrészlet megvizsgálja az átadott hibaüzenetet, és tájékoztatja annak természetéről. Amennyiben a felhasználó futás közben megszakítja a rutint, ez a programrészlet semmilyen eredményt nem ad. Ugyanez a helyzet, ha a programban valamilyen hibafeltétel teljesülésekor az **exit** függvény meghívására kerül sor. Ez feltételezi azt, hogy valamelyik nyomtatási utasítással előzőleg már tudatta a felhasználóval a hiba természetét. Ne felejtse el meghívni a **princ** függvényt, ha azt kívánja, hogy a hibarutin végén visszatérési érték jelenjen meg.

Ha a VISSZA parancs hívásaival biztosítja, hogy a rutint egyetlen lépésben érvényteleníteni lehessen, akkor gondoskodnia kell a VISSZA másodszori meghívásáról, amit normális kilépéskor a rutin maga végezne el. Szükség lehet arra is, hogy az AutoCAD VISSZA parancsával érvénytelenítse mindazt, amit a rutin létrehozott. A programok az UNDOCTL és UNDOMARKS rendszerváltozók segítségével dönthetik el, hogyan hívják meg a VISSZA parancsot, ha ez szükséges egyáltalán.

A hibakezelő rutinokkal kapcsolatban nem szabad megfeledkezni arról, hogy ezek is ugyanolyan AutoLISP függvények, akár a többi, így a felhasználónak módjában áll megszakítani futásukat. Törekedjék arra, hogy a lehető legrövidebbek és leggyorsabbak legyenek. Ezzel

növeli annak valószínűségét, hogy meghívása esetén a teljes rutin lefut.

Sok ADS- és ARX-definiált parancsnak saját hibaüzeneteket tartalmazó függvényei vagy rendszerváltozói vannak. Ezek részletes leírását a 14., "Külsőleg definiált parancsok és rendszerváltozók használata" című fejezetben találja.

A hibákról úgy is értesítheti a felhasználót, ha "figyelmeztető ablakot" jelenít meg a képernyőn. Ez egy kis párbeszédpanel, mely az Ön által írt program üzenetet tartalmazza. A figyelmeztető ablakot az **alert** függvény meghívásával jelenítheti meg.

A következő példában az **alert** függvény egy figyelmeztetõablakot jelenít meg.

```
(alert "A fájl nem található")
```

Erről bővebb leírást a 13. fejezetben, az **"alert"** címszónál talál.

Alkalmazások kezelése

Az AutoLISP rendelkezik alkalmazásokat és külsőleg definiált függvényeket kezelő függvényekkel. Az alkalmazáskezelő függvények a következők:

ads	arxunload	autoxload	xload
arx	autoarxload	load	xunload
arxload	autoload	startapp	

Az alábbi rész a **load**, az **arx**, és a **arxload** függvényekre mutat példákat. "Az automatikus betöltés parancsa" című résznél, a 164. oldalon az **autoload** függvényre talál példát. Minden egyes alkalmazás kezelő függvény leírását megtalálja a 13. , az "AutoLISP függvények katalógusa" című fejezetben.

AutoLISP alkalmazások betöltése

Függvénydefiníciókat *.lsp* kiterjesztésű fájlokban tárolhat ezeket a **load** AutoLISP függvénnyel töltheti be. Amennyiben ezt a fájlt egy *acad.lsp* vagy egy *.mnl* fájlba illeszti, az AutoCAD indításákor az automatikusan betöltődik. A *.lsp* fájlok kifejezései betöltéskor kiértékelődnek. Egy *.lsp* fájl leggyakrabban a **defun** függvényt használja arra, hogy a később végrehajtandó függvénycsoportokat tároljon a memóriában.

S::STARTUP függvény—automatikus végrehajtás

Amennyiben a felhasználó által definiált **S::STARTUP** függvény az *acad.lsp* vagy egy *.mnl* fájlban van, akkor az új vagy már létező rajz megnyitásakor meghívásra kerül. Így az *acad.lsp* fájlban az **S::STARTUP** függvény mellett a **defun** függvényt is megadhatja a beállítások elvégzéséhez.

Amennyiben az AutoCAD KILÉP és VÉGE parancsait a saját változataival kívánja helyettesíteni, olyan *acad.lsp* fájlt használjon, amely a következőket tartalmazza:

```
(defun C:KILÉP ( )
  ... az Ön definíciója ...
)
(defun C:VÉGE ( )
  ...az Ön definíciója ...
)
(defun S::STARTUP ( )
  (command "deftöröl" "kilép")
  (command "deftöröl" "vége")
)
```

A **defun** függvény még a rajz inicializálása előtt újradefiniálja a KILÉP és VÉGE parancsokat. A rajz inicializálása után az **S::STARTUP** függvény kerül meghívásra, a KILÉP és a VÉGE parancsok eredeti definíciója pedig megszűnik.

Mivel egy **S::STARTUP** függvényt sok helyen definiálhat (egy *acad.lsp* fájlban, egy *.mnl* fájlban vagy bármely más AutoLISP fájlban, melyek ezekből a fájlokból töltődnek be), lehetőség van arra, hogy egy előzőleg definiált **S::STARTUP** függvényt felülírjon. A következő eljárással biztosítható, hogy az Ön indítófüggvénye más függvényekkel dolgozzon.

```
(defun MYSTARTUP ( )
  ... az Ön indítófüggvénye ...
)

(if S::STARTUP
  (setq s::startup (append s::startup '((mystartup)) ))
  (defun s::startup ( ) (mystartup) )
)
```

Az előző program hozzáilleszti az indítófüggvényét a már létező **S::STARTUP** függvényhez, és az indítófüggvénnyel együtt újradefiniálja az **S::STARTUP** függvényt. Attól függetlenül, hogy létezik-e az **S::STARTUP** függvény az előző program megfelelően működik.

Betöltési módszerek

Az AutoLISP programok általában *.lsp* fájlok formájában állnak rendelkezésre, amelyeket a **load** függvénnyel lehet betölteni. A

felhasználó számára elérhető függvények definiálásán túlmenően, a **load** művelet arra is lehetőséget nyújt a program készítőjének, hogy a program használatára vonatkozó útmutatást adjon a felhasználónak. Minthogy az AutoLISP kiértékeli a fájlban talált kifejezéseket, egy **princ** (illetve **print**, **prin1** vagy hasonló) függvényhívás beillesztésével tetszőleges információt jeleníthet meg a promptsorban. Ha viszont a fájl legutolsó sora a "**(princ)**", akkor ezzel felülírható az, amit egyébként az AutoLISP megjelenítene.

Egy fájl betöltésekor egyéb függvényhívásokra is lehetőség nyílik. Az AutoLISP az *.lsp* fájlban található minden olyan kifejezést végrehajt a fájl betöltésekor, amely nem egy **defun** utasítás része. Ezt felhasználhatja arra, hogy a már említett szöveges információ megjelenítésén túl beállítson bizonyos paramétereket, vagy végrehajtson bármilyen egyéb kezdeti műveletet.

ADS és ARX alkalmazások betöltése

Mielőtt egy ADS-ben vagy ARX-ben definiált parancsot vagy függvényt egy AutoLISP alkalmazásban meghívna, ellenőrizze, hogy az adott parancs vagy függvény érvényes-e. Az **ads** és az **arx** függvények megadják az éppen betöltött ADS és ARX alkalmazások listáját.

A következő mintaprogram az **INLIST** függvényt definiálja, melynek két argumentuma van. Az első argumentum, 1st, egy karakterlánc-lista, a második argumentum, str, egy karakterlánc. Az **INLIST** függvény visszatérési értéke T, ha az *str* karakterlánc az *lst* listában található karakterláncok egyike. Ezt a függvényt az alkalmazásba illesztve, ellenőrizhei, hogy egy ADS vagy ARX alkalmazás be van-e töltve.

```
(defun INLIST ( lst str / ct tmpstr ret )
  (setq ct 0)
  (repeat (length lst)
    (setq tmpstr (nth ct lst))
    (if (wcmatch tmpstr (strcat "*" str "*"))
      (setq ret T)
    )
    (setq ct (1+ ct))
  )
  ret
)
```

Amennyiben az **INLIST** függvény definiálva van, a következő kód ellenőrzi, hogy a Render ARX alkalmazás be van-e töltve. Amennyiben a Render nincs betöltve, a program betőlti a *render.arx* fájlt. A mintaprogramnak van egy *onfailure* argumentuma (ami hasonlít a korábban leírt **load** függvény által használtra). Ez jelzi a felhasználónak, ha nem találta meg a *render.arx* fájlt. Ez a program az

ads függvény által megadott karakterlánclistát (elérési útvonal és fájlnevek) használja az *lst* argumentumként. Mivel a karakterláncban, az *str* argumentumhoz hasonlóan, a program megkülönbözteti a kis- és nagybetűket, ezért a karakterláncnak az Ön platformja szerinti formátumúnak kell lennie. Azt, hogy a platform tesz-e különbséget kis- és nagybetű között a parancssorban meghívott **arx** vagy **ads** függvénnyel ellenőrizheti. A kis- és nagybetük megkülönböztetése miatt a következő példa csak Windows vagy DOS rendszerben érvényes.

```
(if (not (INLIST (arx) "RENDER") )
  (setq err (arxload "render" nil))
)
(if (not err) (princ "\nHiba: RENDER.ARX nincs
betöltve."))
```

8

ÁLTALÁNOS SEGÉDFÜGGVÉNYEK

Az AutoLISP program különböző függvényeket kínál a pillanatnyilag betöltött rajz tartalmának vizsgálatára. Ez a fejezet bemutatja ezeket a függvényeket és leírja, hogyan használhatók más függvényekkel együtt.

Lekérdező és parancsfüggvények

Az ebben a részben bemutatott lekérdező- és parancsfüggvények közvetlen hozzáférést nyújtanak az AutoCAD parancsokhoz és rajzszolgáltatásokhoz. Viselkedésük az AutoCAD rendszer- és környezeti változók pillanatnyi állapotának és az éppen betöltött rajznak a függvénye.

A lekérdező és parancsfüggvények a következőek:

acad_colordlg	getcfg	getvar	setvar
command	getenv	setcfg	ver

Ebben a részben a **command**, **getcfg**, **getvar**, **setcfg** és **getvar** függvényekre vonatkozó példák találhatóak meg. A **help**, **getenv** és **ver** függvények leírása a 13., „AutoLISP függvények katalógusa" című fejezetben található meg. Az AutoCAD súgórendszeréhez való hozzáférés a **help** függvény segítségével lehetséges.

Parancsok meghívása a command függvénnyel

A **command** függvény az AutoCAD parancsot közvetlenül az AutoCAD Parancs prompthoz küldi. A **command** függvénynek változó hosszúságú argumentumlistája van. Ezen argumentumoknak meg kell felelniük az adott parancs által elvárt típusnak és értékeknek. Ezek lehetnek karakterláncok, valós és egész számok, pontok, rajzelemnevek illetve kiválasztási halmazok nevei. Az adatok, például szögek, távolságok és pontok karakterláncként vagy az adott értékként (egész vagy valós számként illetve pontlistákként) adhatóak át. Az üres karaktersorozat (`""`) megfelel a szóköz beírásának vagy a ↵ billentyű lenyomásának.

A **command** függvény segítségével használható parancsokra vonatkozik néhány korlátozás, ezeket lásd a **"command"** címszónál a 13. fejezetben.

Az alábbi programrészlet a **command** függvény jellemző hívásaira mutat példákat.

```
(command "kör" "0,0" "3,3")
(command "vastagság" 1)
(setq p1  '(1.0 1.0 3.0))
(setq rad 4.5)
(command "kör" p1 rad)
```

Ha ezen függvények meghívásakor az AutoCAD Parancs prompt módban van, akkor a következő műveleteket végzi el:

1 A **command** első hívása a pontokat karaktersorozatként adja át a KÖR parancsnak (egy (0.0,0.0) középpontú és a (3.0,3.0) ponton átmenő kört rajzol).

2 A második hívás egy egész értéket ad át a VASTAGSÁG parancsnak (az aktuális vastagságot 1.0-re változtatja).

3 Az utolsó hívás egy háromdimenziós pontot és egy valós (lebegőpontos) értéket használ, amelyek változóként tárolódnak, és a KÖR parancsnak alapértékként adódnak át (egy másik [kihúzott] kört rajzol (1.0,1.0,3.0) középponttal és 4.5 egység sugárral).

Nemzeti (honosított) nyelv-támogatás

Ha olyan AutoLISP programokat kíván írni, amelyek az AutoCAD honosított változata alatt is futhatnak, akkor a szabványos AutoCAD parancsok és kulcsszavak elé egy aláhúzás karaktert (_) kell tennie, így a rendszer azokat automatikusan lefordítja.

```
(command "_line" pt1 pt2 pt3 "_c")
```

Ha a pont előtagot használja (az újradefiniált parancsok használatának elkerülésére), akkor a pontot és az aláhúzást bármilyen sorrendben kiteheti a parancsnév elé. A `"._line"` és a `"_.line"` sorrend is érvényes.

Várakozás a felhasználói adatbevitelre

Ha egy AutoCAD parancs végrehajtása közben a rendszer a **command** függvény argumentumaként a PAUSE nevű előredefiniált szimbólummal találkozik, akkor a parancs várakozni kezd (végrehajtása felfüggesztődik), hogy lehetővé váljon a közvetlen felhasználói adatbevitel (általában pontkiválasztás vagy mozgatás). Ez hasonló a menüknél alkalmazható visszahajló törtvonal-szünet módszerhez.

Ha a **command** függvény felfüggesztett állapota alatt kiad egy transzparens parancsot, ez nem változtat a függvény állapotán. Ezért a 'ZOOM és a 'TOL használható a **command** felfüggesztett állapotában. A szünet állapot fennmarad mindaddig, amíg az AutoCAD érvényes adatot nem kap, és nincs transzparens parancs érvényben. Például az alábbi kód kiadja a KÖR parancsot, a középpontot az (5,5) pontba állítja, majd várakozni kezd, hogy a felhasználó a képernyőn mozgatással beállíthassa a kör sugarát. Amint a felhasználó megadja a kívánt pontot (vagy begépeli a kívánt sugarat), a függvény továbblép, és húz egy vonalat az (5,5) és a (7,5) pontok között.

```
(command "kör" "5,5" pause "vonal" "5,5" "7,5" "")
```

A menün keresztüli adatbevitel nem szakad meg az AutoLISP szünet hatására. Ha egy menüpont aktív, mialatt a **command** függvény adatbevitelre várva szünetel, az adatbevitelre vonatkozó kérést a menü elfogadhatja. Ha azt akarja, hogy maga a menüpont is fel legyen függesztve, egy visszahajló törtvonal (\) jelet kell beírnia a menüpontba. Amint az érvényes adatbevitel megtörtént, mind a **command** függvény, mind a menüpont továbblép.

Kijelölt pontok átadása az AutoCAD parancsoknak

Néhány AutoCAD parancs (mint például a METSZ, ELÉR és LEKEREKÍT) igényli, hogy a felhasználó az objektumon kívül megadjon egy *kijelölt pontot* is. Ahhoz, hogy az ilyen objektumból és pontból álló adatpárokat a **command** függvény segítségével és a PAUSE használata nélküli átadhassa, először változókként kell azokat tárolnia. A pontok a **command** függvényen belül karakterláncokként, vagy a függvényen kívüli definiálásuk után változókként adhatók át, az alábbi példában látható módon. Ez a programtöredék arra mutat be egy módszert, hogyan lehet átadni egy rajzelemnevet és egy kijelölt pontot a **command** függvénynek.

```
(command "kör" "5,5" "2")          Kört rajzol
(command "vonal" "3,5" "7,5" "")   Vonalat rajzol
(setq el  (entlast))               Megadja az utolsó rajzelem-
                                   nevet
(setq pt  '(5 7))                  Beállítja a pt pontot
(command "metsz" el "" pt "")      Végrehajtja a lemetszést
```

Ha ezen függvények meghívásakor az AutoCAD Parancs prompt módban van, az AutoCAD a következő műveleteket fogja elvégezni:

1. Megrajzol egy (5,5) középpontú és 2 egység sugarú kört.
2. Húz egy vonalat a (3,5) pontból a (7,5) pontba.
3. Létrehoz egy `el` nevű változót, ami az adatbázishoz legutoljára hozzáadott objektum neve. (Bővebben az objektumokról és az objektumkezelő függvényekről a 9., "Kiválasztási halmazokat, objektumokat és szimbólumtáblákat kezelő függvények" c. fejezetben talál).
4. Létrehoz egy `pt` nevű változót, ami egy pont a körön. (Ez a pont adja meg a kör kimetszendő részét.)
5. Végrehajtja a METSZ parancsot az `el` objektum és a `pt` által megadott pont kiválasztásával.

Rendszer- és környezeti változók

A **getvar** és **setvar** függvények teszik lehetővé az AutoLISP alkalmazások számára, hogy az AutoCAD rendszerváltozók értékét lekérdezzék és megváltoztassák. Ezek a függvények a változó nevének megadására egy karakterláncot használnak. A **setvar** függvény egy a rendszerváltozó által elvárt típusú értéket határoz meg. Az AutoCAD rendszerváltozók sokféle típusúak: lehetnek egészek, valós értékek, karakterláncok, két- és háromdimenziós pontok. A **setvar** függvény argumentumaként átadott értékeknek megfelelő típusúaknak kell lenniük. Érvénytelen típus átadása AutoLISP hibaüzenetet eredményez. A rendszerváltozók listáját és azok típusát lásd az AutoCAD on-line súgójában, a „Rendszerváltozók" cím alatt.

Az alábbi programtöredék biztosítja, hogy a következő LEKEREKÍT parancs legalább 1 egység sugarat használjon:

```
(if (< (getvar "filletrad") 1)
  (setvar "filletrad" 1)
)
```

Az AutoLISP rutinok számára a **getenv** nevű függvény biztosít hozzáférést az operációsrendszer pillanatnyilag definiált környezeti változóihoz.

Konfigurációvezérlés

Az AutoCAD az *acad.ini* fájlt használja a konfigurációs információk tárolására. Ezen fájl `Appdata` része teszi lehetővé a felhasználók és fejlesztők számára a saját alkalmazásaikhoz tartozó konfigurációs információ tárolását. A **getcfg** és **setcfg** függvények teszik lehetővé az AutoLISP alkalmazások számára, hogy az `Appdata` rész paramétereinek értékét lekérdezzék és megváltoztassák.

Képernyővezérlés

Az AutoLISP néhány függvénye az AutoCAD képernyő vezérlésére szolgál, beleértve mind a szöveges, mind a grafikus ablakokat. Néhány függvény ezek közül a felhasználói beviteltől függ, illetve arra szólít fel. A képernyővezérlés függvényei a következőek:

graphscr	grvecs	print	textpage
grclear	menucmd	prompt	textscr
grdraw	prin1	redraw	vports
grtext	princ	terpri	

A **terpri** és a **vports** függvények leírása a 13. fejezetben található.

Üzenetek megjelenítése a parancssorban

Az alapvető kimeneti függvények a **prompt**, a **princ**, a **prin1**, és a **print**. A **prompt** függvény egy üzenetet (karakterláncot) ír az AutoCAD parancssorba, és `nil` értékkel tér vissza. A **princ**, a **prin1** és a **print** függvények mind egy kifejezést (nem feltétlenül egy karaktersorozatot) jelenítenek meg a parancssorban, és visszatérési értékük a kifejezés; ezenkívül a kimenetet egy fájlba is tudják küldeni. A különbségek a következőek: a **princ** függvény a karaktersorozatokat a lezáró idézőjelek nélkül jeleníti meg, míg a **prin1** és a **print** idézőjelek közé zárva írja ki azokat; a **print** ezenkívül a kifejezés elé egy üres sort, utána pedig egy szóközt szúr be.

A következő fájlkezelő függvények szintén képesek a kimenetek megjelenítésére a parancssorban (ezekről a függvényekről bővebben a "Fájlkezelés" című résznél, a 254. oldalon találhat információt).

read-char read-line write-char write-line

A **prompt** függvény által megjelenített karakterlánc hossza lehetőleg ne haladja meg a grafikus ablak parancssorának hosszúságát, amely rendszerint nem több 80 karakternél.

Az alábbi példa a négy alapvető kimeneti függvény közötti különbséget szemlélteti, és azt, hogyan kezelik ugyanazt a karakterláncot (az itt látható oktális kód DOS rendszerre érvényes; az Ön rendszerére vonatkozó oktális kódokat az operációs rendszerének kézikönyvében találhatja meg, az alapelv azonban az operációs rendszertől függetlenül azonos). Az AutoLISP számos vezérlőkaraktert biztosít a karakterláncokon belüli használathoz. A teljes listát a "Vezérlőkarakterek a karakterláncokban" című résznél, a 201. oldalon találja.

```
(setq str "A \"megengedett\" dőlésszög: \361 \254\"")
```

(prompt str) *kiírja* A "megengedett" dőlésszög ± 1/4"
és visszaadja nil

(princ str) *kiírja* A "megengedett" dőlésszög ± 1/4"
és visszaadja " A "megengedett" dőlésszög ± 1/4""

(prin1 str) *kiírja* " A "megengedett" dőlésszög ± 1/4""
és visszaadja " A "megengedett" dőlésszög ± 1/4""

(print str) *kiírja* *<blank line>*
" A "megengedett" dőlésszög ± 1/4"" *<space>*
és visszaadja " A "megengedett" dőlésszög ± 1/4""

Menük vezérlése

A grafikus ablakmenük megjelenítését a **menucmd** függvény vezérli. Megjeleníti, módosítja, vagy lekérdezi az aktuális menü egyik almenüjét, és elfogadja annak a karakterláncnak az argumentumát, amely meghatározza az almenüt és a rajta végrehajtandó feladatot. Mivel a DOS és UNIX menüfájlok különböznek Windows alattiaktól, a **menucmd** által végrehajtandó tevékenység típusa attól a platformtól függ, ahol az Ön alkalmazása fut.

A **menucmd** függvény egy karakterlánc argumentummal rendelkezik, amely egyenlőségjellel elválasztott két mezőből áll, a következő módon:

```
"menü_terület=akció"
```

Ez a szintaxis egy almenüt tölt a kiválasztott menüterületre; vagy végrehajt egy tevékenységet a menüponton, illetve az aktuálisan betöltött menüterületen. A *menü_terület* mező meghatározza, hogy a menü melyik része kapja meg a tevékenységet. Ez a mező egy menüterületet is meghatároz, mint például a P0 (a kurzormenü számára) az S (a képernyőmenü számára), vagy egy kiválasztott menüpont. Az *akció* mező meghatározza azt a műveletet, amit a menüterületen, a menüponton, vagy az almenü menüterületre történő töltésénél kell végrehajtania. A menüterületek, amelyek egy műveletet fogadnak el, megegyeznek a menüfájlok almenü referenciáiban használtakkal. (További tudnivalók az "Almenük" című résznél, a 91. oldalon.)

Minden menüterület tartalmaz egy aktuális betöltött almenüt. Alapértelmezés szerint az első almenü után következő menü-szakasz címke ebbe a menüterületbe van betöltve.

Ha a *menü_terület* egy legördülő vagy ábramenüt határoz meg, akkor az *akció* egy csillag (*) is lehet. Ez jeleníti meg a menüt. (A legördülő- és az ábramenük meghívásukkor nem automatikusan jelennek meg). Windows alatt csak a P0 (kurzor) menü és az ábramenük jelennek meg a csillag jel hatására. További információt talál a menüfájlokról, a 4. "Saját menük" című fejezetben.

Megjegyzés A menüfájl hasonló magyarázatait bevezető dollár jelet ne írja be a karakterlánc argumentumba. Ezen kívül a menüfájlban a karakterlánc argumentum *akció* mezőjében lévő, az almenü címkéket megelőző csillag karaktert sem szabad beírni.

Az alábbi **menucmd** függvényhívás hatására az aktuális menüfájlban definiált **OSNAP képernyőalmenü jelenik meg (feltételezve, hogy a képernyőmenü pillanatnyilag engedélyezve van).

```
(menucmd "S=OSNAP")
```

A Windows rendszerben lehetséges a menücsoportra való hivatkozás. Ez hasznos lehet, ha azonos almenüneveket tartalmazó, többszörös menük vannak betöltve. Az alábbi kód az ACAD menücsoport **OSNAP képernyőalmenüjét jeleníti meg. A menücsoportok leírása a "Windows-specifikus menüjellemzők" című résznél, a 127. oldalon található.

```
(menucmd "S=ACAD.OSNAP")
```

A **menucmd** függvény be tud tölteni almenüket a BUTTONS és AUX menüterületekre. Ha az a célja, hogy a digitalizáló eszközének nyomógombjai a Tábla mód be- vagy kikapcsolt állapotától függően különbözőképpen működjenek, definiáljon a ***BUTTONS1 részben két almenüt (**DIG-BUTTONS és **TAB-BUTTONS), és az alábbi kóddal váltson át közöttük:

```
(menucmd "B1=DIG-BUTTONS") DIG-BUTTONS almenü engedélyezve
(menucmd "B1=TAB-BUTTONS") TAB-BUTTONS almenü engedélyezve
```

Az alábbi kód betölti a ***POP0 menüt a P0 (kurzor) menüterületre és megjeleníti azt.

```
(menucmd "P0=POP0")      A ***POP0 menüt a P0 menüterületre tölti
(menucmd "P0=*")         Megjelenítés
```

Ha biztos benne, hogy a megfelelő menü töltődött be egy bizonyos menüterületre, akkor nem kell minden megjelenítés előtt külön betölteni azt.

Az alábbi hívás megjeleníti a P1 (első legördülő menü) területre jelenleg betöltődött legördülő menüt (csak DOS esetén).

```
(menucmd "P1=*")
```

Ha a "P1=*" hivatkozást úgy használja, hogy előzőleg nem töltött be menüt, akkor az bármilyen eredménnyel járhat. Jóllehet, szinte bármilyen menüt betölthet egy legördülő vagy kurzormenü területre, a legjobb megoldást az arra a területre tervezett menük használata jelenti. Például, ha van egy **MORESTUFF nevű almenüje, akkor az alábbi kód segítségével betöltheti azt a P1 területre:

```
(menucmd "P1=MORESTUFF")   A **MORESTUFF menüt a P1 menüterületre tölti
(menucmd "P1=*")           Megjelenítés
```

Ez a menü itt marad mindaddig, amíg egy másik menü betöltésével ki nem cseréli, például így:

```
(menucmd "P1=POP1")
```

Ha a menü használja a letiltó és kijelölő lehetőségeket, akkor a **menucmd** függvény segítségével visszakeresheti és megváltoztathatja a menücímke állapotát. Az alábbi hívás a P2 nevű legördülő menü negyedik címkéjének pillanatnyi állapotát adja meg.

```
(menucmd "P2.4=#?")      Ha tiltott a "P2.4=~" értékkel tér vissza
```

Ezek a függvényhívások aktívvá teszik, majd letiltják ugyanazt a címkét:

```
(menucmd "P2.4=")        Engedélyezi a címkét
(menucmd "P2.4=~")       Letiltja a címkét
```

Ezen kívül a menücímkék bal oldalára jelöléseket helyezhet el, vagy azokat törölheti onnan.

A menüpontok kezelésének fentiekben leírt módja viszonylag jól működik egy egyszerű statikus menü esetén (mint a DOS és UNIX rendszerekben). Ezzel szemben megbízhatatlanná válik, ha (például a Windows rendszerben) a menüpontok helye többszörös részmenü-fájlok betöltésekor megváltozik. Windows rendszerben használja a menücsoport és névcímke lehetőségeket a menüpontok kezelésére.

Ahelyett, hogy egy menüpontot a menüfájl-beli helye alapján határozna meg, a hozzá kapcsolt menücsoportot és névcímkét adja meg.

Ha a menücímkék ki- illetve bekapcsolásához és megjelöléséhez a menücsoport lehetőséget használja, akkor a csoportnév elé egy G karaktert kell írnia az alábbi példákon látható módon:

```
(menucmd "Gacad.ID_New=~")  Letiltja a címkét
(menucmd "Gacad.ID_New=")   Engedélyezi a címkét
```

Egy AutoLISP függvény nemcsak a menücímkék ki- és bekapcsolására képes, hanem a címkében kijelzett szöveget is megváltoztathatja úgy, hogy a címkében egy DIESEL karakteres kifejezést helyez el. Mivel a DIESEL csak karaktersorozatokat fogad el bemenő adatként, a DIESEL kifejezésnek egy olyan USERS1–5 rendszerváltozón keresztül továbbíthat információt, amely a használt függvény által visszaadott értékre van beállítva.

A **menucmd** függvényt AutoLISP függvényen belüli DIESEL karakteres kifejezések kiértékelésére is használhatja. Az alábbi rutin megadja az aktuális időt:

```
(defun C:CTIME ( / ctim)
  (setq ctim
    (menucmd "M=$(edtime,$(getvar,date),H:MMam/pm)"))
  (princ (strcat "\nA pontos idő: " ctim ))
  (princ)
)
```

A DIESEL kifejezések AutoLISP környezetben való használatára vonatkozó további információkat és a DIESEL függvények felsorolását megtalálja az 5., „ Állapotsor-konfiguráció és DIESEL karakteres kifejezésnyelv" c. fejezetben.

A grafikus és szöveges ablakok vezérlése

A grafikus és szöveges ablakok megjelenítése AutoLISP alkalmazásokból vezérelhető. Egyképernyős AutoCAD konfigurációk esetén, a **graphscr** függvényt hívja meg, ha a grafikus, és a **textscr** függvényt, ha a szöveges képernyőt kívánja megjeleníteni. Ezen függvények használata egyenértékű a Képváltó billentyű használatával.
A **textpage** függvény a **textscr** függvényhez hasonló, azonban a **textpage** a megjelenítés előtt törli a szöveges képernyőt (ahogyan az AutoCAD STÁTUSZ és LISTA parancsa teszi).

A **redraw** függvény az AutoCAD FRISSÍT parancsához hasonló, azonban további beavatkozást tesz lehetővé a megjelenítésben: nemcsak a teljes grafikus képernyő felfrissítésére használható, hanem egyetlen rajzelem újrarajzolására vagy eltüntetésére (azaz a háttérszínnel történő kirajzolására) is. Olyan összetett rajzelemek esetében, mint egy Vonallánc vagy egy Blokk, az egész rajzelem, vagy

csak annak fejléce is felfrissíthető (vagy eltüntethető). A **redraw** függvény segítségével egy adott rajzelem normál illetve kiemelt állapotba hozható.

Alacsonyszintű grafika vezérlése

Az AutoCAD rendelkezik olyan alacsonyszíntű grafikus vezérlésre alkalmas függvényekkel, amelyek közvetlen hozzáférést biztosítanak az AutoCAD grafikus képernyőjéhez és bemeneti eszközeihez.

A **grclear** függvény letörli az aktuális nézetablakot, anélkül, hogy a grafikus ablak prompt-, státusz- és menüterületeire hatással lenne. A **grtext** függvény közvetlenül a státusz- illetve menüterületeken jelenít meg szöveget, kiemeléssel vagy anélkül. A **grdraw** függvény egy vektort rajzol az aktuális nézetablakba, és beállítja a vektort normál illetve kiemelt állapotban jeleníti meg. A **grvecs** függvény több vektort rajzol.

Megjegyzés Ezek a függvények szoros kapcsolatban állnak az AutoCAD belső kódjával, ezért működésük várhatóan verzióról verzióra változni fog. Ezért nincs rá garancia, hogy az ezeket a függvényeket használó alkalmazások felfelé kompatibilisek lesznek. Emellett e függvények az aktuális hardverkonfigurációtól is függenek: különösen a **grtext** függvényhívást tartalmazó alkalmazások esetében valószínű, hogy az egyes konfigurációkon eltérően működnek, hacsak a fejlesztő nem elég óvatos, és nem a leírtak szerint jár el , kerülve a hardverspecifikus szolgáltatások használatát. (Lapozza fel a 4., "Saját menük" című fejezetet.) Végezetül, minthogy ezek alacsony szintű függvények, szinte semmilyen hibajelzést nem adnak, és váratlanul megváltoztathatják a grafikus képernyő megjelenését (Tekintse meg az alábbi példát ennek rendbehozatalára).

Az alábbi sorrendben meghívott két függvény helyrehozza a **grclear**, **grtext**, **grdraw**, vagy **grvecs** függvények helytelen hívása következtében a grafikus képernyőn mutatkozó esetleges hibákat:

```
(grtext)          Visszaállítja a szabvány szövegeket
(redraw)
```

Felhasználói bevitel kérése

Az AutoLISP alkalmazások számos függvény segítségével kérhetnek adatbevitelt az AutoCAD felhasználójától. A felhasználói bevitelre szolgáló függvények a következők:

entsel	getfiled	getpoint	nentsel
getangle	getint	getreal	nentselp
getcorner	getkword	getstring	

getdist getorient initget

A getxxx függvények

A **getxxx** beviteli függvények mindegyike egy megfelelő típusú adat bevitelét várja a felhasználótól, majd a bevitt értékkel tér vissza. Az alkalmazásban opcionálisan egy prompt is megadható, amelyet a függvény megjelenít, mielőtt várakozni kezd. A következő táblázat tartalmazza a **getxxx** függvényeket és a hozzájuk tartozó felhasználói bevitel típusát.

Megengedett beviteli módok a getxxx felhasználói bevitelre szolgáló függvényekhez

Függvénynév	*A felhasználói bevitel típusa*
`getint`	Egy egész számérték a promptsorban megadva
`getreal`	Egy valós vagy egész számérték a promptsorban megadva
`getstring`	Egy karakterlánc a promptsorban megadva
`getpoint`	Egy pont a promptsorban megadva, vagy a képernyőn kijelölve
`getcorner`	Egy pont (egy ablak átellenes sarka) a promptsorban megadva, vagy a képernyőn kijelölve
`getdist`	Egy valós vagy egész számérték (egy távolság) a promptsorban megadva, vagy a képernyőn kijelölt pontok által meghatározva
`getangle`	Egy szögérték (az aktuális szögformátumban) a promptsorban megadva, vagy a képernyőn kijelölt pontok által meghatározva
`getorient`	Egy szögérték (az aktuális szögformátumban) a promptsorban megadva, vagy a képernyőn kijelölt pontok által meghatározva
`getkword`	Egy előre meghatározott kulcsszó vagy annak rövidítése a promptsorban megadva

Megjegyzés Bár a **getvar**, **getcfg** és **getenv** függvények is a *g, e,* és *t* betűkkel kezdődnek, ezek nem felhasználói bevitelre szolgáló függvények. Bővebb ismertetést találhat a "Lekérdező és parancsfüggvények" című részné, a 224. oldalon.

A **getint**, **getreal**, és **getstring** függények mindegyike az AutoCAD promptsorából várja a megfelelő típusú felhasználói bevitelt. Csak a kért típusú értékkel térnek vissza.

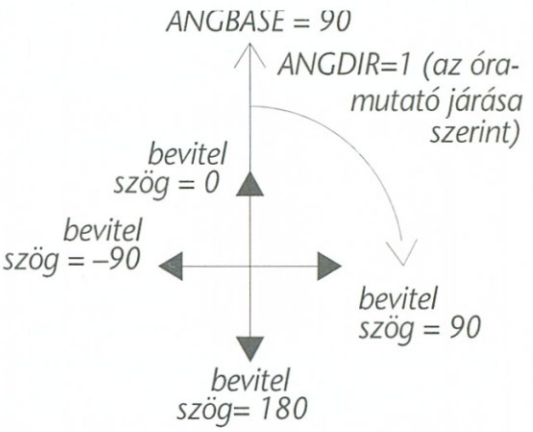
A **getpoint**, **getcorner**, és **getdist** függvények a felhasználói bevitelt a promptsorba írt értékek, vagy a képernyőn kijelölt pontok formájában várják. A **getpoint** és a **getcorner** függvények háromdimenziós pontokkal, a **getdist** függvény pedig egy valós számértékkel tér vissza.

A **getangle** és a **getorient** függvény egyaránt a promptsorba beírt, vagy a grafikus képernyőn pontkijelöléssel meghatározott szögérték megadását várja. A **getorient** függvény számára a nulla szög mindig a jobbra, azaz "keletre" vagy "3 óra felé" mutató iránynak felel meg. A **getangle** számára a nulla szöget az ANGBASE rendszerváltozó értéke jelenti, amely bármilyen szögértékre beállítható. A **getangle** és **getorient** függvények egyaránt egy szögértékkel (valós számmal) térnek vissza, amely radiánban értendő, az óramutató járásával ellentétes irányban, attól a nulla irányszögtől mérve, amely a **getangle** esetében az ANGBASE értékével egyenlő, a **getorient** esetében pedig jobbra mutat.

Tételezzük fel, hogy az ANGBASE 90 fokra (Észak) van beállítva, az ANGDIR értéke pedig 1 (a szögek az óramutató járásával egyező irányban növekszenek). A következő táblázat a **getangle** és a **getorient** függvény visszatérési értékeit mutatja (radiánban) a bemenet nevezetes szögértékei esetén.

A getangle és a getorient néhány lehetséges visszatérési értéke

Bevitel (fokban)	*getangle*	*getorient*
0	0.0	1.5708
–90	1.5708	3.14159
180	3.14159	4.71239
90	4.71239	0.0



A **getangle** függvény figyelembe veszi az ANGDIR és az ANGBASE rendszerváltozó beállítását, amikor elfogadja a bevitelt. A **getangle** függvényt használhatja tehát egy beilleszteni kívánt blokk elforgatási szögének megadásához, mivel nulla fokos bemeneti szögérték esetén a függvény mindig nulla radiánt ad vissza. A **getorient** függvény csak

az ANGDIR rendszerváltozót veszi figyelembe. Ezért a **getorient** függvény olyan szögek megadására alkalmas, mint egy szöveges objektum alapvonalának iránya. Például az ANGBASE és az ANGDIR fenti beállítása mellett, egy nulla elforgatási szöggel létrehozott szöveg alapvonala esetében a **getorient** függvény 90 foknak megfelelő szögértékkel tér vissza.

A felhasználói bevitelre szolgáló függvények kihasználják az AutoCAD hibaellenőrző képességét. A nyilvánvaló hibákat az AutoCAD észleli, és a hibás értéket a beviteli függvény nem fogadja el. Az **initget** függvény (tárgyalása később) előzetes hívásával további szűrők iktathatók be, így kisebb mértékben szükséges a külön hibaellenőrzés.

A **getkword** függvény egy kulcsszót, vagy annak rövidítését várja bemenetként. A kulcsszavakat a **getkword** hívása előtt az **initget** függvény segítségével kell definiálni. Valamennyi felhasználói bevitelre szolgáló függvény (a **getstring** kivételével) az alapvetően várt értéken kívül kulcsszavakat is elfogadhat, feltéve, hogy azok definiálására előzőleg az **initget** függvény hívásával sor került.

Minden beviteli függvény lehetővé teszi egy opcionális *prompt* argumentum megadását. Egy alkalmazásban a felhasználói bevitel kéréséhez inkább ennek az argumentumnak a használatát javasoljuk, mintsem egy azt megelőző **prompt** vagy **princ** függvényhívást. Ha ugyanis a beviteli függvény hívásában szerepel a *prompt* argumentum, akkor érvénytelen felhasználói bevitel esetén ez a prompt újból megjelenik. Amennyiben a *prompt* argumentum nincs megadva, és a felhasználó hibás választ ad, akkor az AutoCAD promptsorában a következő üzenet jelenik meg:

Próbálja meg újból:

Ez zavaró lehet, mivel elképzelhető, hogy az eredeti prompt már nem látható a Parancs promptterületen.

Megjegyzés A felhasználó egy beviteli függvényre általában *nem* válaszolhat egy AutoLISP kifejezés bebillentyűzésével. Ha azonban Ön egy AutoLISP rutinban az **initget** függvényt alkalmazza, akkor bizonyos függvények esetében tetszőleges billentyűzet-bemenetet engedhet meg, ami az AutoLISP felhasználásával definiált parancsok számára is lehetővé teszi AutoLISP kifejezések válaszként történő elfogadását. Ezt a "Tetszőleges bevitel a billentyűzetről" című résznél, a 237. oldalon" találja.

A felhasználói bevitelre szolgáló függvények feltételeinek szabályozása

Az **initget** függvény biztosítja a következő beviteli függvények meghívásának különböző szintű vezérlését. Az **initget** függvény a következő **entsel**, **nentsel**, **nentselp**, vagy **getxxx** (a **getstring**, **getvar**, és **getenv** kivételével) függvény számára különböző opciókat

állít be. A függvénynek két argumentuma lehet, a `bits` és a `string`, melyek bármelyike el is hagyható. A `bits` argumentum egy vagy több szabályzóbitet határoz meg, amelyek bizonyos bemeneti értékeket engedélyeznek vagy tiltanak a következő felhasználói bevitelre szolgáló függvény hívása esetén. A `string` argumentumban kulcsszavak adhatók meg, melyeket a következő felhasználói bevitelre szolgáló függvényhívás elfogad.

Az **`initget`** függvény által beállított szabályzóbitek és kulcsszavak csak a következő beviteli függvényhívásra vonatkoznak, azt követően azonnal automatikusan érvényüket vesztik. Az alkalmazásban nincs szükség az **`initget`** ismételt hívására a speciális feltételek törléséhez.

A felhasználói bevitelre szolgáló függvények bemeneti opciói

A `bits` argumentum értéke a következő beviteli függvényhívásra adható válasz adattípusait korlátozza. Ezáltal kevesebb hibaellenőrzéssel is elérhető, hogy a felhasználó csak a kívánt típusú információt adhassa meg. Íme néhány a bitek lehetséges beállításai közül: 1 — nem engedélyez üres inputot, 2 — nem engedélyez 0 (nulla) értéket, 4 — nem engedélyez negatív értéket. Ha ezek a bitértékek egy ezt követő **`getint`** függvényhívásra vonatkoznak, akkor a felhasználó csak egy nullánál nagyobb egész számot adhat meg.

Több feltétel egyidejű beállításához egyszerűen adja össze az értékeket (tetszőleges kombinációban), így a `bits` értéke 0 és 255 között lesz. Amennyiben a `bits` argumentum nem szerepel, vagy értéke nulla, akkor a következő beviteli függvényhívásra a szabályzó feltételek egyike sem vonatkozik. A **`initget`** lehetséges bitbeállításainak teljes felsorolása a 13. fejezetben, az "**`initget`**" címszó alatt található.

```
(initget (+ 1 2 4))
(getint "\nHány éves? ")
```

A fenti programsorok a felhasználó életkorát kérdezik meg. Az AutoCAD egy hibaüzenetet jelenít meg és megismétli a promptot, ha a felhasználó megkísérel negatív értékkel vagy nullával válaszolni, esetleg csupán a ↵ billentyűt nyomja le, vagy egy karakterláncot ír be (maga a **`getint`** függvény elvet minden olyan próbálkozást, amely egész számtól eltérő érték megadására irányul).

Kulcsszó opciók

Az opcionális `string` argumentum olyan kulcsszavakból álló listát határoz meg, melyeket a következő felhasználói beviteli függvényhívás elfogad.

Az **`initget`** függvény a teljes kulcsszavak mellett rövidítések használatát is megengedi. A felhasználói bevitelre szolgáló függvény az előre definiált kulcsszavak egyikével tér vissza, ha a bevitel betű

szerint megegyezik a kulcsszóval (tekintet nélkül a kis- és nagybetűkre), vagy a felhasználó a kulcsszó rövidítését billentyűzi be. A kulcsszavak rövidítésére két módszer áll rendelkezésre, mindkettőt a "Kulcsszavak meghatározása" című résznél, a 465. oldalon tárgyaljuk.

A következő, felhasználó által definiált függvényben a **getreal** hívását egy **initget** hívás előzi meg, amely két kulcsszót definiál. Az alkalmazás figyeli ezeket a kulcsszavakat, és a beviteli értéket ennek megfelelően állítja be:

```
(defun C:GETNUM (/ szám)
  (initget 1 "Pi Két-pi")
  (setq szám (getreal "Pi/Két-pi/<szám>: "))
  (cond
    ((eq szám "Pi") pi)
    ((eq szám "Két-pi") (* 2.0 pi))
    (T szám)
  )
)
```

Ez az **initget** függvényhívás megakadályozza az üres bevitelt (*bits* = 1), valamint létrehozza a "Pi" és "Két-pi" kulcsszavakból álló listát. Ezután a **getreal** függvényhívás egy valós szám megadását kéri a következő prompt megjelenítésével:

Pi/Két-pi/szám:

Az eredményt a num lokális szimbólum fogja tartalmazni. Amennyiben a felhasználó megad egy számot, a **C:GETNUM** függvény ezzel az értékkel tér vissza. Ha azonban a felhasználó a Pi (vagy egyszerűen csak P) kulcsszót billentyűzi be, akkor a **getreal** a Pi értékkel tér vissza. A **cond** függvény ezt felismeri, és ebben az esetben π értékével tér vissza. A Két-pi kulcsszó kezelése hasonló.

Megjegyzés Az **initget** függvény használatával lehetővé teheti, hogy válaszként az **entsel**, **nentsel**, és **nentselp** függvények is elfogadjanak kulcsszavakat. Ezekről a függvényekről további információt talál az "Objektumkezelés" című résznél, a 272. oldalon, valamint a függvények ismertetésénél a 13. fejezetben az **"entsel,"** **"nentsel,"** és **"nentselp"** címszavaknál.

Tetszőleges bevitel a billentyűzetről

Az **initget** függvény a legtöbb **getxxx** függvény számára tetszőleges billentyűzet-bevitelt is megenged. Ezt a beviteli értéket az alkalmazás karakterláncként kapja meg. Így olyan alkalmazás írható, amely megengedi, hogy a felhasználó AutoLISP kifejezést hívjon meg válaszul egy **getxxx** függvény promptjára.

Ezek a függvények egy olyan módszert mutatnak be, amely AutoLISP választ enged meg egy **getxxx** függvényhívásra:

```
(defun C:ARBENTRY ( / pt1)
  (initget 128)                          ;Beállítja a tetszőleges beviteli
                                          bitet.
  (setq pt1 (getpoint "\nPoint: "))      ;Értéket vár a felhaználótól.
  (if (= 'STR (type pt1))                ;Ha az egy karakterlánc
    (setq pt1 (eval (read pt1)))         ;szibólummá konvertálja és
                                         ;megpróbálja a függvényt
                                         ;kiértékelni
pt1                                      ;Egyébként visszaadja az értéket.
  )
)

(defun REF ( )
  (setvar "LASTPOINT" (getpoint "\nReferencia pont: "))
  (getpoint "\nKövetkező pont: " (getvar "LASTPOINT"))
)
```

Ha a **C:ARBENTRY** és a **REF** függvény betöltése is megtörtént, a következő parancssorozat elfogadható.

Parancs: **arbentry**
Pont: **(ref)**
Referenciapont: *Válasszon egy pontot*
Következő pont: **@1,1,0**

A bevitel érvényessége

Védje meg beállításait a véletlen felhaszálói hibáktól. Az AutoLISP **getxxx** felhasználói beviteli függvénye ezt elvégzi Ön helyett. Azonban ügyeljen arra, hogy lehetnek más programmal is összefüggő beállítások, melyeket a **getxxx** függvény nem ellenőriz. Amenyiben a bemenet érvényességének ellenőrzését nem végzi el, ez komolyan befolyásolhatja a program egységét.

Geometriai segédfüggvények

A függvények egy csoportja lehetővé teszi az alkalmazások számára, hogy hozzájussanak a rajzok geometriai adataihoz, illetve a rajzokra vonatkozó geometriai információhoz. A geometriai segédfüggvények a következők:

angle	inters	polar
distance	osnap	textbox

A **distance** függvény két pont távolságát állapítja meg; az **angle** függvény egy egyenes és az *X* tengely (az aktuális FKR *X* tengelye) által bezárt szöget méri radiánban; a **polar** függvény pedig

polárkoordináták segítségével határoz meg egy pontot (egy kiindulóponthoz viszonyítva). Az **inters** függvény két egyenes metszéspontját keresi meg. Az **osnap** és a **textbox** függvényeket külön tárgyaljuk.

A következő programrészlet a geometriai segédfüggvények néhány egyszerű hívását mutatja be:

```
(setq pt1 '(3.0 6.0 0.0))
(setq pt2 '(5.0 2.0 0.0))
(setq bázis '(1.0 7.0 0.0))
(setq radiánok  (angle pt1 pt2))    Az aktuális FKR XZ síkjában fekvő szög (radiánban mért eredmény)
(setq len  (distance pt1 pt2))      távolság a háromdimenziós térben

(setq endpt  (polár bázis radiánok hossz))
```

A **polar** függvény hívásának eredményeképpen az `endpt` változó értéke az a pont lesz, amely az (1,7) ponttól ugyanolyan távolságban és irányban van, mint a `pt1` a `pt2` ponttól.

Tárgyraszter

Az **osnap** függvény az AutoCAD Tárgyraszter módjainak egyikével talál meg egy pontot. A tárgyraszter módok karakterlánc típusú argumentumokkal határozhatók meg.

A következő esetben az **osnap** függvény egy rajzelem felezőpontját keresi a `pt1` pont közelében:

```
(setq pt2  (osnap pt1 "fel"))
```

A következő esetben egy objektum `pt1`-hez legközelebbi felezőpontját, végpontját vagy középpontját keresi:

```
(setq pt2 (osnap pt1 "fel,vég,köz"))
```

A `pt2` változó mindkét példában azzal a ponttal lesz egyenlő, amely eleget tesz a tárgyraszter követelménynek. Ha nem csak a raszterpontok elégítik ki ezt, a pont kiválasztása a SORTENTS rendszerváltozó beállításától függ. Egyébként akkor `pt2` értéke `nil` lesz.

Megjegyzés A Tárgyraszter használatakor az APERTURE rendszerváltozó határozza meg azt, hogy a megadott pontnak milyen közel kell lennie egy objektumhoz.

A szöveg kiterjedése

A **textbox** függvény egy szövegobjektum köré rajzolható keret átlós koordinátáit adja eredményül. Egyetlen argumentuma egy olyan

rajzelem-definíció típusú lista, mint amilyet az **entget** függvény szolgáltat (csoportkódok és értékek asszociációs listája). Ez a lista lehet a szövegobjektmot definiáló teljes asszociációs lista, vagy ennek csupán a szöveget leíró részlete.

A **textbox** függvény által visszaadott pontok a szövegobjektumot határoló keretet (egy a szövegelemet körbezáró képzeletbeli ablak) úgy határozzák meg, mintha a szöveg beillesztési pontja a (0,0,0) pont, elforgatási szöge pedig 0 volna. Az első helyen visszaadott lista általában a (0.0 0.0 0.0) pont, hacsak a szövegobjektum nem dőlt, nem függőleges, vagy nem tartalmaz olyan karaktereket, melyeknek alsó szára az alapvonal alá nyúlik (mint például a *g* vagy a *p* betű). Ez az első lista a szöveg köré rajzolható legkisebb téglalap bal alsó sarokpontjának a szöveg beillesztési pontjához viszonyított koordinátáit adja meg. A második lista ennek a téglalapnak a jobb felső sarokpontját határozza megg. A függvény által visszaadott pontok mindig ennek a befoglaló keretnek a bal alsó és jobb felső sarokpontját határozzák meg, tekintet nélkül a vizsgált szöveg irányára.

Ebben a példában a **textbox** függvény által még elfogadott minimális elemdefiníciós lista látható. Mivel egyéb információ nem áll rendelkezésre, a **textbox** függvény az aktuális alapértelmezés szerinti szövegstílust és magasságot veszi figyelembe.

Parancs: **(textbox '((1 . "Helló világ")))**
((0.0 0.0 0.0) (2.80952 1.0 0.0))

A következő példa azt mutatja be, hogyan lehet a **textbox** függvény számára egy elemdefiníciós listát átadni.

Parancs: **dszöveg**
Igazít/Stílus/<Kezdőpont>: **1,1**
Magasság <1.0000>: ↵
Elforgatási szög <0>: ↵
Szöveg: **test**
Szöveg: ↵
Parancs: **(setq e (entget (entlast)))**
((-1 . <Rajzelem neve: 6000001c>) (0 . "TEXT") (8 . "0")
(10 1.0 1.0 0.0) (40 . 1.0) (1 . "test") (50 . 0.0)
(41 . 1.0)(51 . 0.0) (7 . "STANDARD") (71 . 0) (72 . 0)
(11 0.0 0.0 0.0) (210 0.0 0.0 1.0) (73 . 0))
Parancs: **(textbox e)**
((0.0 0.0 0.0) (0.8 0.2 0.0))

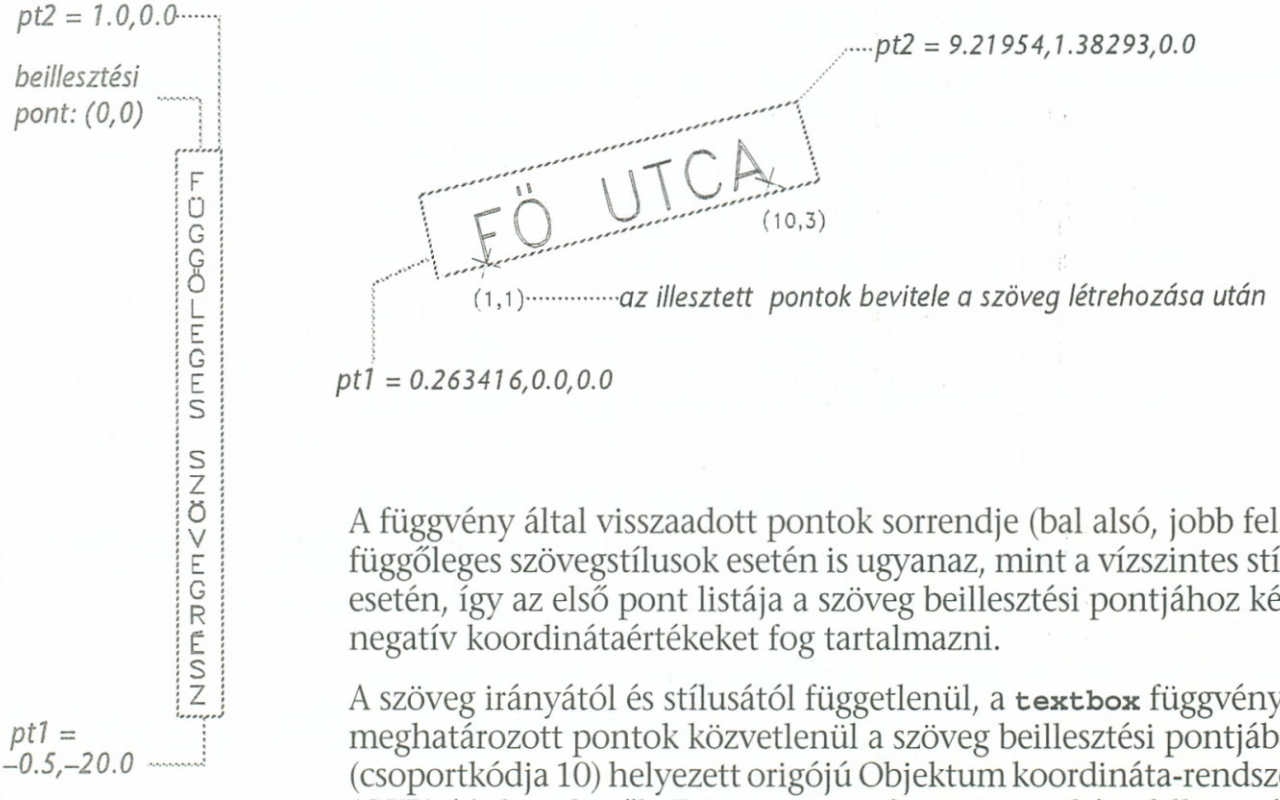
A következő ábra egy 1.0 magasságértékkel rendelkező szövegobjektum esetében mutatja a **textbox** függvény eredményét. Az ábrán a szöveg alapvonala és beillesztési pontja is látható.

A textbox függvény által meghatározott pontok

A `pt1` és `pt2` pontok függőleges vagy elforgatott helyzetű szöveg esetében is megtartják egymáshoz viszonyított helyzetüket (bal - jobb, lent - fent). A bal alsó pont koordinátái szükség esetén negatív értékek is lehetnek.

A következő ábra függőleges és illesztett mintaszövegek esetében mutatja a **`textbox`** függvény által meghatározott pontok (`pt1` és `pt2`) értékét. A betűk magassága mindkét minta esetében 1.0 értékű (az illesztett szöveg magassága az illesztési pontoknak megfelelően változik).

Függőleges és illesztett szöveg



A függvény által visszaadott pontok sorrendje (bal alsó, jobb felső) a függőleges szövegstílusok esetén is ugyanaz, mint a vízszintes stílusok esetén, így az első pont listája a szöveg beillesztési pontjához képest negatív koordinátaértékeket fog tartalmazni.

A szöveg irányától és stílusától függetlenül, a **`textbox`** függvény által meghatározott pontok közvetlenül a szöveg beillesztési pontjába (csoportkódja 10) helyezett origójú Objektum koordináta-rendszerben (OKR) értelmezhetők. Ezt a pontot referenciapontként felhasználva, a **`textbox`** által szolgáltatott koordináták a szöveg tényleges kiterjedését meghatározó pontokká transzformálhatók. Az alábbiakban két rutint mutatunk be mintául arra, hogy a **`textbox`** függvény segítségével hogyan lehet keretet rajzolni egy kiválasztott szöveg köré, függetlenül annak elhelyezésétől.

Ez a rutin keretet rajzol egy kiválasztott szöveges objektum köré a **textbox** függvény felhasználásával.

```
(defun C:TBOX ( / textent tb ll ur ul lr)
  (setq textent (car (entsel "\nVálasszon ki egy szöveget:
")))
  (command "fkr" "Objektum" textent)
  (setq tb (textbox (list  (cons -1 textent)))
        ll (car tb)
        ur  (cadr tb)
        ul  (list (car ll) (cadr ur))
        lr (list (car ur) (cadr ll))
  )
  (command "vlánc" ll lr ur ul "Zár")
  (command "fkr" "p")
  (princ)
)
```

A következő rutin ugyanazt a feladatot hajtja végre, mint az előző, az AutoLISP **sin** és **cos** függvényeivel végzett geometriai számítások útján. Az eredmény csak abban az esetben lesz helyes, ha az aktuális FKR párhuzamos a szövegobjektum síkjával.

```
(defun C:TBOX2 ( / textent ang sinrot cosrot
                   t1 t2 p0 p1 p2 p3 p4)
  (setq textent (entget (car (entsel "\nVálasszon egy
szöveget: "))))
  (setq p0 (cdr (assoc 10 textent))
       ang (cdr (assoc 50 textent))
       sinrot (sin ang)
       cosrot (cos ang)
       t1 (car (textbox textent))
       t2 (cadr (textbox textent))
       p1 (list
          (+ (car p0)
            (- (* (car t1) cosrot)(* (cadr t1) sinrot))
          )
          (+ (cadr p0)
            (+ (* (car t1) sinrot)(* (cadr t1) cosrot))
          )
       )
       p2 (list
          (+ (car p0)
            (- (* (car t2) cosrot)(* (cadr t1) sinrot))
          )
          (+ (cadr p0)
            (+ (* (car t2) sinrot)(* (cadr t1) cosrot))
          )
       )
```

```
        p3 (list
           (+ (car p0)
              (- (* (car t2) cosrot)(* (cadr t2) sinrot))
           )
           (+ (cadr p0)
              (+ (* (car t2) sinrot)(* (cadr t2) cosrot))
           )
        )
        p4 (list
           (+ (car p0)
              (- (* (car t1) cosrot)(* (cadr t2) sinrot))
           )
           (+ (cadr p0)
              (+ (* (car t1) sinrot)(* (cadr t2) cosrot))
           )
        )
    )
    (command "vlánc" p1 p2 p3 p4 "z")
    (princ)
)
```

Konverziók

Az ebben a részben ismertetett függvények adattípusok és mértékegységek konvertálására szolgálnak. A konverziós függvények az alábbiak:

angtof	atof	cvunit	rtos
angtos	atoi	distof	trans
ascii	chr	itoa	

A konverziós függvényekről a 13. fejezetben is olvashat.

Karakterlánc-konverziók

Az **rtos** (valósból karakterlánc) és **angtos** (szögből karakterlánc) függvények az AutoCAD által használt számértékeket konvertálják karakterlánc-értékké, amely kimenetben vagy szöveges adatként használható fel. Az **rtos** függvény egy valós számértéket, az **angtos** pedig egy szöget alakít át. Az eredményül kapott karakterlánc formátumát az AutoCAD rendszerváltozói szabályozzák: valós (lineáris) értékek esetében a mértékegységeket és a pontosságot a LUNITS és a LUPREC, szögértékek esetében pedig az AUNITS és az AUPREC rendszerváltozó határozza meg. Mindkét függvény esetében a DIMZIN méretváltozó adja meg, hogy a számok elején illetve végén álló nullák bekerüljenek-e az eredményül kapott karakterláncba.

A következő programrészletek az **rtos** függvény néhány hívását és visszatérési értékét mutatják be (feltételezve, hogy a DIMZIN változó

értéke nulla). A pontosság (az **rtos** harmadik argumentuma) az első hívásban 4, a többiben 2 helyiérték.

```
(setq x 17.5)
(setq str "\nAz érték az adott formátumban: ")

(setq fmtval  (rtos x 1 4))          Mód1 1 = tudományos
(princ (strcat str fmtval))          eredménye  Az érték formátuma: 1.7500E+01

(setq fmtval (rtos x 2 2))           Mód 2 = tizedes
(princ (strcat str fmtval))          eredménye  Az érték formátuma: 17.50

(setq fmtval (rtos x 3 2))           Mód 3 = mérnöki
(princ (strcat str fmtval))          eredménye  Az érték formátuma: 1'-5.50"

(setq fmtval (rtos x 4 2))           Mód 4 = építészeti
(princ (strcat str fmtval)           eredménye  Az érték formátuma: 1'-5 1/2"

(setq fmtval (rtos x 5 2))           Mód  5 = tört
(princ (strcat str fmtval))          eredménye  Az érték formátuma: 17 1/2
```

Amennyiben a UNITMODE rendszerváltozó értéke 1, vagyis az értékek ugyanolyan alakban jelennek meg, mint ahogyan a megadásuk történik, akkor az **rtos** függvény által visszaadott karakterlánc a mérnöki (*mód* értéke 3), építészeti (*mód* értéke 4) és tört (*mód* értéke 5) mértékegység esetén eltér a fentiektől. Például, az előzőekben bemutatott kimeneti formátumok közül az első kettő változatlan marad, az utolsó három sor azonban a következőképpen jelenik meg:

```
Az érték formátuma: 1'5.50"
Az érték formátuma: 1'5-1/2"
Az érték formátuma: 17-1/2
```

Mivel az **angtos** függvény figyelembe veszi az ANGBASE rendszerváltozót, a következő kifejezés mindig "0" eredményt ad.

```
(angtos (getvar "angbase"))
```

Nincs olyan AutoLISP függvény, amely akár az ANGBASE elforgatási szögét a valódi nullához (Kelethez) képest, akár egy tetszőleges szögnek a radiánban kifejezett értékét karakterlánc formájában adná vissza.

Ha meg kívánja tudni egy tetszőleges szög méretét, vagy az ANGBASE elforgatási szögét az AutoCAD nulla irányához (Kelet) képest, a következő lehetőségek közül választhat:

- A kérdéses szöget adja hozzá az ANGBASE aktuális értékéhez, és vizsgálja meg, hogy az eredmény abszolút értéke nagyobb-e, mint 2π. Ha igen, akkor vonjon le belőle 2π-t (vagy adjon hozzá 2π-t, ha az eredmény negatív), majd ennek eredményét jelenítse meg az **angtos** függvénnyel.

- Az ANGBASE értékét átmenetileg helyezze el egy változóban, és állítsa az ANGBASE értékét nullára. Így értékelje ki az előbbi értéket az **angtos** függvénnyel, majd állítsa vissza az ANGTOS eredeti értékét.

Az `(atof (angtos 0))` kifejezés eredményét 360 fokból (2π radiánból vagy 400 grádból) kivonva, szintén megkaphatja az ANGBASE nullához viszonyított elforgatási szögét.

A **distof** (distance to floating point) függvény az **rtos** fordítottja, így az alábbi függvényhívások, melyek az előző példákban előállított karakterláncokat használják fel, egységesen a 17.5 értéket adják eredményül (figyelje meg a visszahajló törtjel alkalmazását a 3-as és 4-es mód esetén).

```
(distof "1.7500E+01" 1)      Mód 1 = tudományos

(distof "17.50" 2)           Mód 2 = tizedes

(distof "1'-5.50\"" 3)       Mód 3 = mérnöki

(distof "1'-5 1/2\"" 4)      Mód 4 = építészeti

(distof "17 1/2" 5)          Mód 5 = tört
```

A következő programrészletek az **angtos** függvény hasonló hívásait és visszatérési értékeit mutatják be (továbbra is feltételezve, hogy a DIMZIN értéke nulla). A pontosság (az **angtos** harmadik argumentuma) az első hívásban 0, az azt követő háromban 4, az utolsóban pedig 2 helyiérték.

```
(setq ang 3.14159 str2 "\nA szög az adott formátumban ")

(setq fmtval (angtos ang 0 0))     Mód 0 = tizedes fok
(princ (strcat str2 fmtval))        eredménye  A szög formátuma 180

(setq fmtval (angtos ang 1 4))     Mód 1 = fok/perc/másodperc
(princ (strcat str2 fmtval))        eredménye  A szög formátuma 180d0'0"

(setq fmtval (angtos ang 2 4)      Mode 2 = grad
(princ (strcat str2 fmtval))       eredménye  A szög formátuma 200.0000g

(setq fmtval (angtos ang 3 4)      Mód 3 = radián
(princ (strcat str2 fmtval))        eredménye  A szög formátuma  3.1416r

(setq fmtval (angtos ang 4 2)      Mód 4 = geodéziai
(princ (strcat str2 fmtval))        eredménye  A szög formátuma W
```

Amennyiben az **angtos** függvény geodéziai mértékegységben adja vissza a karakterláncot (a *mód* értéke 4), a UNITMODE rendszerváltozó szintén befolyásolja az eredményt. Ha a UNITMODE nullával egyenlő, akkor az eredményül kapott karakterlánc szóközöket is tartalmazhat

(például "É 45f K"). Ha a UNITMODE értéke 1, akkor a karakterlánc nem tartalmaz szóközöket (például "É45fK").

Az **angtof** függvény az **angtos** fordítottja, így az összes alábbi függvényhívás a 3.14159 értéket adja eredményül.

```
(angtof "180" 0)          Mód 0 = fok
(angtof "180d0'0\"" 1)    Mode 1 = fok/perc/másodperc
(angtof "200.0000g" 2)    Mód 2 = grad
(angtof "3.14159r" 3)     Mód 3 = radiáns
(angtof "W" 4)            Mód4 = geodéziai
```

Amennyiben egy karakterláncban a távolságot láb és hüvelyk mértékegységben, vagy a szöget fokokban, percekben és másodpercekben fejezi ki, akkor a visszahajló törtjelet kell az idézőjel előtt használnia (\"), megkülönböztetve azt a karakterláncot lezáró idézőjeltől. Ezt az **angtof** és a **distof** függvény előző példái is szemléltetik.

Szögkonverziók

Amennyiben szögek értékeit kívánja radiánból fokra átszámolni, használja az **angtos** függvényt, ami egy karakterláncot ad vissza. Ezután az **atof** függvény segítségével a karakterláncot lebegőpontos értékké konvertálja .

```
(setq rad 1.5708)
(setq deg (atof (angtos rad 0 2)))      értéke    90.0
```

Még hatásosabb lehet, ha egy dtr függvényt illeszt az alkalmazásba. A dtr a fokokat számolja át radiánra. A következő programrészlet ezt mutatja be.

```
; Fokból radiánba konvertálja a szöget
(defun DTR (d)
  (* pi (/ d 180.0))
)
```

Miután ezt a függvényt definiálta, a következő módon használhatja a **DTR** függvényt alkalmazásaiban:

```
(setq deg (dtr rad))                    értéke    90.0
```

ASCII kódok konverziója

Az AutoLISP az **ascii** és a **chr** függvényekkel kezeli a decimális ASCII kódokat. Az **ascii** függvény egy, a megfelelő karakterlánchoz tartozó, ASCII decimális értéket ad meg, a **chr** függvény pedig egy ASCII decimális értékhez tartozót karaktert jelenít meg.

Amennyiben látni kívánja rendszerének összes karakterét, valamint azok decimális, oktális és hexadecimális értékeit, mentse el a következő AutoLISP programot *ascii.lsp* néven. Ezután töltse be a fájlt,

és adja meg az új ASCII parancsot a Parancs promptban. A parancs az ASCII kódokat a képernyőre, valamint a *ascii.lsp* nevű fájlba írja. A **C:ASCII** függvény egy másik függvényt használ, a **BASE** függvényt. Ezt a konverziós segédprogramot valószínűleg más alkalmazásban is hasznosnak találja majd.

```
; A BASE függvény a decimális egészeket más módon alakítja
karakterláncokká
(defun base ( bas int / ret yyy zot )
  (defun zot ( i1 i2 / xxx )
    (if (> (setq xxx (rem i2 i1)) 9)
      (chr (+ 55 xxx))
      (itoa xxx)
    )
  )
  (setq ret (zot bas int) yyy (/ int bas))
  (while (>= yyy bas)
    (setq ret (strcat (zot bas yyy) ret))
    (setq yyy (/ yyy bas))
  )
  (strcat (zot bas yyy) ret)
)

(defun C:ASCII ( / kar ki ct code dec okt hex )
  (initget "Igen")
  (setq chk (getstring "\nÍrás a  ASCII.TXT fájlba, folytassam? <I>:
"))
  (if (or (= chk "Igen")(= chk ""))
    (progn
      (setq out (open "ascii.txt" "w") chk 1 code 0 ct 0)
      (princ "\n \n KAR   DEC  OKT  HEX \n")
      (princ "\n \n KAR   DEC  OKT  HEX \n" out)
      (while chk
        (setq dec (strcat "   " (itoa code))
          oct (base 8 code) hex (base 16 code))
        (setq dec (substr dec (- (strlen dec) 2) 3))
        (if (< (strlen oct) 3)(setq oct (strcat "0" oct)))
        (princ (strcat "\n " (chr code) "      " dec "   "
          oct "   " hex ) )
        (princ (strcat "\n " (chr code) "      " dec "   "
          oct "   " hex ) out)
```

```
        (cond
          ((= code 255)(setq chk nil))
          ((= ct 20)
            (setq xxx (getstring
              "\n \nNyomja le az 'X' billenytűt a kilépéshez, vagy
bármi mást a folytatáshoz: "))
            (if (= (strcase xxx) "X")
              (setq chk nil)
              (progn
                (setq ct 0)
                (princ "\n \n KAR   DEC   OKT   HEX \n")
              )
            )
          )
        )
        (setq ct (1+ ct) code (1+ code))
      )
      (close out)
      (setq out nil))
    )
  )
  (princ)
)
```

Mértékegységek átalakítása

Az *acad.unt* fájl különböző konverziókat definiál olyan valós mértékegységek között, mint a mérföld/kilométer, Fahrenheit/Celsius, és stb. A **cvunit** függvény az egyik rendszerben kifejezett értéknek a másik rendszerben vele egyenlő értékével tér vissza. A két mértékegység-rendszert az *acad.unt* fájlban definiált mértékegységeket tartalmazó karakterláncok határozzák meg

A **cvunit** függvény nem alakítja át az egymással inkompatibilis dimenziókat. Centimétert például nem alakít grammá.

Amikor a **cvunit** függvény első ízben számít át egy mértékegységet egy adott rajzszerkesztési alkalom során, meg kell keresnie az adott mértékegységet definiáló karakterláncot az *acad.unt* fájlban. Ha olyan alkalmazást készít, amelyben a mennyiségeket sokszor kell átszámítani egyik mértékegység-rendszerből a másikba, akkor sokkal hatékonyabb, ha az 1.0 értéket számítja át a **cvunit** függvény egyszeri hívásával, majd az eredményt mint arányossági tényezőt használja az ezt követő átszámításokban. Ez a módszer az *acad.unt* fájlban definiált valamennyi mértékegység esetében alkalmazható, kivéve a hőmérsékleti skálákat, amelyek átszámítása az aránytényezővel történő szorzáson kívül egy összeadást is magában foglal.

Konverzió hüvelykről méterre

Amennyiben az aktuális rajz mérnöki vagy építészeti egységeket tartalmaz (láb és hüvelyk), a következő rutin a felhasználó által hüvelykben megadott távolságokat méterre konvertálja át.

```
(defun C:I2M ( / eng_len metric_len eng metric)
  (princ "\nHüvelyk konvertálása méterre. ")
  (setq eng_len
     (getdist "\nÍrja be a hosszúságot hüvelykben: "))
  (setq metric_len  (cvunit eng_len "hüvelyk" "méter"))
  (setq eng (rtos eng_len 2 4)
       metric (rtos metric_len 2 4))
   (princ
     (strcat "\n\t" eng " hüvelyk = " metric " méter."))
  (princ)
)
```

A mértékegység-definíciós fájl

Az AutoCAD mértékegység-definíciós fájlja, az *acad.unt* segítségével, olyan tényezőket adhat meg, amelyek lehetővé teszik mértékegységek átszámítását. Az *acad.unt* fájlban a definíciók ASCII formátumúak, és a **cvunit** konverziós függvény használja őket.

Új mértékegységeket is elérhetővé tehet úgy, hogy egy szövegszerkesztővel az *acad.unt* fájlba új definíciókat ír. A definicíók két sorból állnak - az első a *mértékegység neve*, a másik a *mértékegység definíciója*. Az első sor első oszlopában egy csillag jelnek (*) kell lennie, melyet a mértékegység neve követ. Egy mértékegység nevét többféleképpen lehet írni, illetve rövidíteni; ezeket vesszők választják el. Amennyiben a mértékegységnek egyes illetve többes számú alakja is van, azt a következőképpen adhatja meg.

```
*[ [normál] [ ( [egyes.] többes) ] ]...
```

Többtagú kifejezéseket is megadhat (egyes illetve többes számban). A többtagú kifejezések tagjainak nem kell a szó végén kezdődniük, és a többes számú alakot sem kötelező megadni.

```
*inch(es)
*milleni(um.a)
*f(oot.eet) vagy (foot.feet)
```

A * *mértékegység neve* vonalat követő vonal azt adja meg, hogy a mértékegység az *alap*mértékegység vagy *bevezetett* mértékegység.

Alapmértékegységek

Egy alapmértékegység nem más, mint egy konstansokból álló kifejezés. Amennyiben a * *mértékegység név* vonalat követő vonal nem egyenlőségjellel (=) kezdődik, alapmértékegységről van szó. Az alapegység öt egész számot és két valós számot tartalmaz a következőképpen:

```
c, e, h, k, m, r1, r2
```

Az öt egész szám a következő öt konstanshoz tartozik.

c a fény sebessége vákuumban

e az elektron töltése

h Planck-állandó

k Boltzman-állandó

m az elektron nyugalmi tömege

Ezen összetevők együttesen a mértékegység dimenzióit határozzák meg; a hosszúságot, a tömeget, az időt, a sebességet, stb.

Az első valós szám (r1) egy szorzótényező, a másik (r2) pedig egy hozzáadott eltolási érték (csak hőmérséklet-konverzió esetén használatos). Az alapmértékegység-definíció lehetővé teszi a mértékegységek különböző írásmódú változatainak használatát (például *meter* és *metre*), és nem tesz különbséget kis és nagy betűk között. A következő példa egy alapmértékegység-definíciót ad meg.

```
*meter(s),metre(s),m
-1,0,1,0,-1,4.1214856408e11,0
```

Ebben a példában az egy métert meghatározó konstansok:

$$\left(\frac{1}{c}\times h\times\frac{1}{m}\right)\times(4.1214856\ \times\ 10^{11})$$

Származtatott egységek

A származtatott mértékegységek más mértékegységek által definiált mértékegységek. Amennyiben a **mértékegység név* vonalat követő vonal egyenlőségjellel (=) kezdődik, származtatott mértékegységről van szó. Ezekben a kifejezésekben a következő műveletek érvényesek: * (szorzás), / (osztás), + (összeadás), - (kivonás) és ^ (hatványozás). Megadhat korábban definiált mértékegységet úgy, hogy megnevezi azt, és használhat rövidítést is (amennyiben létezik ilyen). A formulán belüli egységek összeszorzódnak , hacsak nem adott meg más aritmetikai műveletet. Például a mértékegység adatbázis definiálja a dimezió nélküli szorzás és osztás neveket. Így a **mikrohüvelyk** beírásával megadhat egy mikrohüvelyk nevű mértékegységet. A következő példák bevezetett mértékegység-definiciók.

```
;Területegységek

*városméret
=93239571.456 méter^2
```

A városméret definiciója 93.239.571,456 négyzetméterként van defiiálva.

```
; Elektromágneses egységek

*volt,v
=watt/amper
```

Ebben a példában a volt definiciója: watt osztva amperrel. Az *acad.unt* fájlban a watt és az amper alapmértékegységként van megadva.

Felhasználói megjegyzések

Amennyiben megjegyzéseket kíván tenni, a sort kezdje pontosvesszővel. A megjegyzés a sor végéig tart.

```
; Ez az egész sor egy megjegyzés
```

Listázza ki az *acad.unt* fájlt, ha több információra illetve példára van szüksége.

Koordináta-rendszerek közötti transzformációk

A **trans** függvény egy pontot vagy elmozdulást transzformál át egyik koordináta-rendszerből a másikba. Első argumentuma a *pt*, mely háromdimenziós pontként, és háromdimenziós elmozdulás-vektorként is értelmezhető, a *disp* argumentumtól függően. Abban az esetben, ha a *disp* elmozdulási argumentum egy nullától különböző érték, akkor a *pt* argumentumot a függvény elmozdulásvektorként, egyébként pedig pontként kezeli. A *from* argumentum határozza meg azt a koordináta-rendszert, amelyben a *pt* értelmezve van, a *to* argumentum pedig a kívánt koordináta-rendszert jelöli ki. Az alábbiakban a **trans** függvény szintaxisát mutatjuk be:

```
(trans pt from to [disp])
```

A következő AutoCAD koordináta-rendszereket lehet a *from* és a *to* argumentumokkal meghatározni.

VKR Világ koordináta-rendszer: a "referencia" koordináta-rendszer. Minden más koordináta-rendszer a VKR-hez képest kerül definiálásra, amely sohasem változik. A VKR-hez viszonyított értékek az egyéb koordináta-rendszerek változása közben is állandók maradnak.

FKR Felhasználói koordináta-rendszer: a "munka" koordináta-rendszer. A felhasználó azért határoz meg egy FKR-t, hogy bizonyos rajzolási feladatokat egyszerűbbé tegyen. Az AutoCAD parancsok számára megadott valamennyi pont, beleértve az AutoLISP rutinok és külső függvények által átadottakat is, az aktuális FKR-ben értendő (hacsak a felhasználó a Parancs promptra adott válaszában nem tesz eléjük egy * jelet). Ha az alkalmazásában VKR, EKR vagy

KKR-beli koordinátákat kíván az AutoCAD parancsainak átadni, akkor ehhez először az FKR-be kell transzformálnia azokat a **trans** függvény segítségével.

OKR Objektum koordináta-rendszer. Az **entget** függvény által szolgáltatott pontok ebben a magához a rajzelemhez viszonyított koordináta-rendszerben értendők. Az ilyen pontokat a VKR, vagy az aktuális FKR illetve KKR-be szokás konvertálni, a rajzelem tervezett felhasználása szerint. Megfordítva, az **entmod** vagy **entmake** függvények útján az adatbázisba beírni kívánt pontokat előzőleg OKR-be kell transzformálni. Az OKR Rajzelem koordináta-rendszer néven is ismert.

KKR Képernyő koordináta-rendszer. Az objektumokat a képernyőn történő megjelenítésük előtt az AutoCAD ebbe a koordináta-rendszerbe transzformálja. A KKR origója a TARGET rendszerváltozóban tárolt pont, *Z* tengelye pedig a nézőirány. Más szóval, egy nézetablak mindig a KKR síknézete. Ezekkel a koordinátákkal meghatározható, mi hol jelenik meg a felhasználó számára.

Abban az esetben, ha a `from` és `to` a kódszámok értéke 2 illetve 3, bármilyen sorrendben, akkor a 2 az aktuális modelltér nézetablakának KKR rendszerét, a 3 pedig a papírtér KKR-t (PTKKR) jelöli. Ha a 2-es kódszámot nem a 3-assal együtt használja (vagy más módon adja meg a koordináta-rendszert), az az aktuális tér — akár papírtér, akár modelltér — KKR-t jelenti, és a másik argumentum szintén az aktuális térben jelöl egy koordináta-rendszert.

PTKKR Papírtér KKR. Ezt a koordináta-rendszert csakis a modelltér éppen aktív nézetablakának KKR-ébe lehet transzformálni, és viszont. Ez lényegében kétdimenziós transzformáció, amely az *X* és *Y* koordinátákat mindig megszorozza egy aránytényezővel, valamint el is tolja azokat, amennyiben az elmozdulási argumentum nulla. A *Z* koordinátát szintén megszorozza az aránytényezővel, azonban sohasem tolja el, így az felhasználható a két koordináta-rendszer közötti aránytényező kiszámítására. A PTKKR (kódszáma 2) csak a modelltér aktuális nézetablakába transzformálható: ha a `from` argumentum értéke 3, akkor a `to` argumentum értéke 2 kell legyen, és fordítva.

A `from` és a `to` argumentum egyaránt egy koordináta-rendszert jelöl, az alábbi módszerek bármelyikével:

- Egy egész típusú kódszámmal, amely a VKR-t, az aktuális FKR-t vagy az aktuális KKR-t (akár aktuális nézetablak, akár papírtér) jelöli.
- Egy rajzelem nevével, amelyet az elemneveket vagy halmazokat kezelő (a 9. Kiválasztási halmaz, Objektumokat és Szimbólumtáblákat kezelő függvények című fejezetben ismertetett) függvények egyike szolgáltatott. Ez az adott rajzelem EKR-ét határozza meg. Síkbeli rajzelemek esetében az EKR különbözhet a VKR-től, mint azt az *AutoCAD Felhasználói kézikönyve* ismerteti. Amennyiben az OKR nem különbözik, az OKR és a VKR közötti konverzió nem változtat az értékeken.
- Egy háromdimenziós kihúzási vektor segítségével szintén meghatározható egy rajzelem EKR-e. A kihúzási vektorok mindig Világkoordinátákat képviselnek. A (0,0,1) kihúzási vektor magát a VKR-t jelöli.

A következő táblázat tartalmazza a `to` és `from` argumentumok által használt, érvényes, egész típusú kódokat.

Koordináta-rendszer kódok

Kód	*Koordináta-rendszer*
0	Világ (VKR)
1	Felhasználói (aktuális FKR)
2	Képernyő: A 0 vagy 1 kóddal együtt használva az aktuális nézetablak KKR-e A 3 kóddal együtt használva a modelltér aktuális nézetablakának KKR-e
3	A Papírtér KKR-e, PTKKR (csak a 2 kóddal együtt használható)

A következő példa egy pontot transzformál a VKR-ből az aktuális FKR-be:

```
(setq pt '(1.0 2.0 3.0))
(setq cs_from 0)                    VKR
(setq cs_to 1)                      FKR
(trans pt cs_from cs_to 0)          disp = 0 jelzi, hogy pt egy pont
```

Amennyiben az aktuális FKR a VKR *Z* tengelye körüli 90 fokos, az óramutató járásával ellentétes irányú elforgatásával jött létre, a **`trans`** függvény a (2.0,-1.0,3.0) ponttal tér vissza. Ha viszont megcseréli a `to` és `from` értékeket, az eredmény az alábbi módon változik:

```
(trans pt cs_to cs_from 0)    az eredmény:  (-2.0,1.0,3.0)
```

Pontok transzformációja

Ha a **trans** függvény segítségével ponttranszformációkat végez és a programnak ezen részét gyorsítania kell, akkor létrehozhatja a saját transzformációs mátrixát úgy, hogy az AutoLISP rendszerben a **trans** függvény egyszeri meghívásával minden egyes „bázisvektort" (0 0 0), (1 0 0) és (0 1 0) transzformál. Mátrixszorzó-függvények írása az AutoLISP programban nehézségekbe ütközhet, így nem éri meg a fáradságot, hacsak a program nem végez sok transzformációt.

Fájlkezelés

Az AutoLISP számos fájlkezelő, illetve adatok bevitelére szolgáló függvényt kínál. A fájlkezelő függvények az alábbiak:

close	help	read-line	write-line
findfile	open	setfunhelp	
getfiled	read-char	write-char	

A **findfile**, **getfiled** és **help** függvények működésére ebben a fejezetben találhatók példák. A többi fájlkezelő függvényt a 13. fejezet tárgyalja.

Fájlkeresés

Az alkalmazások a **findfile** függvényt használják egy adott nevű fájl megkeresésére. A függvény számára kijelölhető, hogy melyik könyvtárban keresse a fájlt, vagy használhatja az aktuális AutoCAD könyvtár-útvonalat is.

A következő programrészletben a **findfile** függvény a kért fájlnevet az AutoCAD könyvtár-útvonalán keresi:

```
(setq refname "refc.dwg")
(setq fil  (findfile refname))
(if fil
  (setq refname fil)
  (princ (strcat "\nA " refname " nevű fájl nem található.
" ))
)
```

Amennyiben a **findfile** hívása sikeres, a `refname` változó értéke a teljes elérési útvonallal megadott fájlnevet tartalmazó karakterlánc lesz, mint például:

```
"/otthon/munka/ref/refc.dwg"
```

Megjegyzés Egy DOS elérési útvonal megadásakor a visszahajló törtjelet (" \ ") meg kell kettőznie ahhoz, hogy azt az AutoLISP elfogadja. Másik lehetőségként a normál törtjel (" / ") is használható a könyvtárak tagolására.

A **getfiled** függvény egy párbeszédpanelt jelenít meg, amely az adott könyvtárban lévő, adott kiterjesztéssel rendelkező fájlok felsorolását tartalmazza. Ez az AutoLISP rutinok számára is biztosítja az AutoCAD fájlkiválasztó párbeszédpanelének elérését.

A **getfiled** függvényhívásban négy argumentum szerepel, melyek a párbeszédpanel megjelenését és működését határozzák meg. Az alkalmazásban a következő karakterláncokat kell megadni: egy cím, amely a párbeszédpanel tetejére kerül; az alapértelmezés szerinti fájlnév, amely a párbeszédpanel alján lévő szerkesztőablakban jelenik meg, valamint a kiterjesztés, amely a listaablakban kezdetben felkínált választékot szabja meg. Ezek bármelyike szükség szerint `nil` is lehet. Az utolsó argumentum egy egész szám, amely azt határozza meg, hogyan működjön együtt a párbeszédpanel a kiválasztott fájlokkal.

Ez az egyszerű rutin a könyvtárszerkezet megtekintését és egy fájl kiválasztását teszi lehetővé a **getfiled** függvény felhasználásával:

```
(defun C:DDIR ( )
  (setq dfil (getfiled "Könyvtár tartalma" "" "" 2))
  (princ (strcat "\nA kiválasztott 'dfil' fájlnév az " dfil "
változóban van))
  (princ)
)
```

Ez a segédparancs alkalomadtán nagyon hasznos lehet. A kiválasztott fájlnevet tartalmazó `fn` változót ezután más AutoLISP függvények is használhatják, vagy az válaszként átadható a parancssorban egy fájlnevet váró promptnak. Ez utóbbi célból a promptra a **!dfil** választ billentyűzheti be.

Megjegyzés A **!dfil** kifejezés párbeszédpanelben nem használható, csak a parancssorban érvényes.

További információ található a 13. fejezet **"getfiled"** címszava alatt.

Súgófájlok elérése

A **help** függvény hozzáférést biztosít mind az AutoCAD (*.ahp*), mind a Windows (*.hlp*) súgófájlokhoz. A súgófájl kiterjesztésétől függően a **help** függvény meghívja az AutoCAD vagy a Windows súgókezelőjét az adott fájllal együtt. Ez a függvény használható arra, hogy saját alkalmazásait súgó szolgáltatással egészítse ki. Az alábbi kódtöredék meghívja az alapértelmezésbeli AutoCAD súgófájlt, és információt szolgáltat a VONAL parancsról.

```
(help "" "vonal")
```

Létrehozhat olyan súgófájlt is, amely információt nyújt alkalmazásairól vagy a munkája során használt eljárásokról. Az alábbi, felhasználó által definiált parancs megjeleníti a *morehelp.ahp* nevű súgófájlt.

```
(defun C:MYHELP ( )
  (help "morehelp.ahp")
  (princ)
)
```

A súgófájlok létrehozását és módosítását a "Saját súgófájlok" című résznél, a 29. oldalon találja. A `setfunhelp` függvény súgót biztosít a felhasználó által definiált parancsok számára. Az új parancs definíciója után elhelyezett a **setfunhelp** függvénynek szóló hívás, egy meghatározott súgófájlt és tárgyszót rendel a parancshoz. Az alábbi példa a *morehelp.ahp* fájlban található Mycmd nevű súgó tárgyszót rendeli a felhasználó által definiált MYCMD parancshoz.

```
(defun C:MYCMD ( )
  .
  .                                    Parancsdefiníció
  .
)
(setfunhelp c:mycmd "morehelp.ahp" "mycmd")
```

Az eszközök elérése és vezérlése

Az AutoLISP rendszerben megfelelő függvények állnak rendelkezésre a különböző beviteli eszközökből származó adatok elérésére. Az eszközhozzáférési és az eszközvezérlő függvények az alábbiak:

grread tablet

Az alábbi fájlkezelő függvények a billentyűzetpufferből származó bemeneti adatokat is tudják olvasni. (Ezekről a függvényekről további információt talál a "Fájlkezelés" című résznél, a 254. oldalon.)

read-char read-line

Felhasználói adatbevitel elérése

A **grread** függvény a felhasználó által bevitt, a billentyűzetről vagy a mutatóeszközről (egér vagy digitalizáló) származó „nyers" adatokat adja vissza. Ha a **grread** függvény hívása engedélyezi a nyomkövetést, a függvény egy digitalizált koordinátaértékkel tér vissza, amely például mint vontatási pozíció használható fel.

Megjegyzés Arra nincs garancia, hogy a **grread** függvényt meghívó alkalmazások felfelé kompatibilisek lesznek. Mivel ez az adott hardverkonfiguráció függvénye, nem valószínű, hogy a **grread** függvényt meghívó alkalmazások minden konfiguráció esetén azonos módon viselkednek .

A digitalizáló táblák kalibrálása

A felhasználó a TÁBLA parancs segítségével kalibrálhatja a digitalizáló táblát (További tudnivalók az AutoCAD on-line súgójában). A **tablet** függvény lehetővé teszi, hogy az alkalmazások közvetlenül is beállíthassák a kalibrációt, és a beállításokat későbbi felhasználás céljából elmentsék.

A **tablet** függvény első argumentuma egy egész típusú *kód*. Amennyiben a *kód* nullával egyenlő, a függvény az aktuális kalibrációval tér vissza. Ha a *kód* értéke 1, a függvény beállítja a kalibrációt a többi argumentumnak megfelelően. A kalibrációt négy, háromdimenziós pont fejezi ki (a *kód* argumentumon kívül). E pontok közül három —a *sor1*, a *sor2* és a *sor3* — a tábla transzformációs mátrixának három sorát jelenti. A negyedik pont —az *irány* — egy vektor, amely merőleges arra a síkra, amelyben a tábla felülete alapértelmezés szerint fekszik (VKR-ben értelmezve). A táblának a TÁBLA paranccsal történő kalibrálása esetén a tábla felülete alapértelmezés szerint az aktuális FKR *XY* síkjában fekszik.

Megjegyzés A TABMODE rendszerváltozó szabályozza, hogy a Tábla mód Be (1) vagy Ki (0) van-e kapcsolva. Ennek értékét a **setvar** függvény segítségével állíthatja be.

Az alábbi rutin lekérdezi a tábla aktuális kalibrációját, és azt a `tcal` szimbólumban helyezi el.

```
(defun C:TELÉRÉS ( )
  (setq tcal (tablet 0))
  (if tcal
    (princ
      (strcat "\nA kalibráció elmentve, "
        "visszaállítására használja a TBEÁLL
parancsot.")
    )
    (princ "\nA kalibráció nem olvasható ki ")
  )
  (princ)
)
```

Amennyiben a fenti rutin végrehajtása sikeres, a `tcal` szimbólum a **tablet** függvény által visszaadott listát tartalmazza. Ez a lista a következő lehet:

```
(1 (0.00561717 -0.000978942 -7.5171)
  (0.000978942 0.00561717 -9.17308)
  (0.0 0.0 1.0)
```

```
    (0.0 0.0 1.0)
)
```

A kalibrációt a **C:TBEÁLL** rutin segítségével állíthatja vissza az előbbi rutin által szolgáltatott értékekre:

```
(defun C:TBEÁLL ( )
  (if (not  (apply 'tablet tcal))
    (princ "\nA kalibráció nem állítható vissza. ")
    (progn
      (princ "\nA tábla kalibrációja visszaállítva. ")
      (setvar "tabmode" 1)
      (if (= (getvar "tabmode") 0)
        (princ "\nA Tábla mód nem állítható be ")
      )
    )
  )
  (princ)
)
```

A *sor1*, *sor2* és *sor3* pontok formájában átadott 3x3-as transzformációs mátrix egy kétdimenziós pont transzformálására szolgál. A kétdimenziós pont oszlopvektorként, *homogén koordinátákkal* fejezhető ki (harmadik elemként az 1.0 értékkel egészül ki), így a transzformáció egyenlete a következőképpen írható fel:

$$\begin{bmatrix} X' \\ Y' \\ D' \end{bmatrix} = \begin{bmatrix} M_{00} & M_{01} & M_{02} \\ M_{10} & M_{11} & M_{12} \\ M_{20} & M_{21} & 1.0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1.0 \end{bmatrix}$$

Egy pont koordinátáinak kiszámítása hasonlóan történik, mint háromdimenziós esetben. Az AutoCAD a pontot a következő képletek szerint transzformálja:

$$X' = M_{00}X + M_{01}Y + M_{02}$$

$$Y' = M_{10}X + M_{11}Y + M_{12}$$

$$D' = M_{20}X + M_{21}Y + 1.0$$

Az így előállított vektorból úgy lesz újra kétdimenziós pont, hogy az első két összetevőjét elosztjuk a harmadikkal, a D′ arányténye zővel, és így megkapjuk a (X′/D′,Y′/D′) pontot.

Projektív transzformáció esetén, ami a legáltalánosabb, a **tablet** függvény gyakorlatilag teljes egészében elvégzi a számítási műveletet. affin és ortogonális transzformáció esetén viszont az M_{20} és az M_{21} egyaránt nulla, így a D′ értéke 1.0 lesz. A D′ kiszámítása és az osztás

elmarad; az eredmény egyszerűen az (X',Y') koordinátájú kétdimenziós pont.

Amint az előző bekezdés is utal rá, az affin transzformáció a projektív transzformáció speciális, uniform esete. Az ortogonális transzformáció az affin transzformáció egy speciális esete, amikor nemcsak hogy M_{20} és M_{21} értéke nulla, hanem az $M_{00} = M_{11}$ és $M_{10} = -M_{01}$ egyenlőségek is fennállnak.

Megjegyzés A kalibráció beállításakor a függvény által visszaadott lista *nem* egyezik a megadott listával, ha az `irány` normalizált. Az AutoCAD ugyanis normalizálja az irányvektort, mielőtt visszaadná az eredményt. Arról is gondoskodik, hogy a harmadik oszlop harmadik elemének (`sor3[Z]`) értéke mindenképpen 1 legyen. Ez a helyzet nem állhat elő, ha a kalibrációt az AutoCAD rendszertől a **`tablet`** függvény útján lekérdezett értékek felhasználásával állítja be. Bekövetkezhet viszont abban az esetben, ha az Ön által írt program maga számítja ki a transzformációt.

Az ASE, az AutoLISP felhasználói felülete

Az AutoLISP amellett, hogy közvetlen hozzáférési lehetőséget nyújt az alkalmazások számára az ASE parancskészlethez (Ismertetése a 14., "Külső definiálású parancsok és rendszerváltozók használata" c. fejezetben található.), egy belső elérést is nyújt az ASE parancsok funkcióinak kihasználásához. Ez felhasználói felületet biztosít az AutoLISP számára az ASE végrehajtható moduljaihoz. Az ASE AutoLISP program felhasználói felülete az alábbi lehetőségeket nyújtja:

- Hozzáférés az ASE párbeszédfelület aktuális beállításaihoz
- Hozzáférés a csatolási információkhoz

Adattípusok

Ez a rész azokat az általános adattípusokat mutatja be, amelyeket az ASE AutoLISP felhasználói felület egyes függvényei argumentumként, vagy visszatérési értékként használnak.

Adatbázis objektumnév —*obj_name*

behatárolt_azonosító*|szabályos_*azonosító

Az ASE program az adatbázis objektumneveket behatárolt és szabályos azonosítókként támogatja. További tudnivalók az *AutoCAD Developers Guide* „Syntax of SQL2 Implementation" című,C függelékében.

Csatolási útvonalnév—*lp_name*

Ez egy csatolási útvonalnév-karaktersorozat, amelynek meg kell felelnie az AutoCAD által a blokknevekkel szemben támasztott követelményeknek.

A csatolási útvonalnevek keresztreferenciákhoz kapcsolódnak, a különböző csatolási útvonalak pedig olyan előtagokkal vannak ellátva, amilyeneket az AutoCAD a blokknevek esetében használ. Az AutoCAD programmal ellentétben a hasonló csatolási útvonalak előtagok hozzáadása nélkül kapcsolódnak.

Hivatkozás adatbázis-objektumra — *obj_path*

```
[kör_név.[kat_név.[séma_név.[tábla_név
[lp_név)]]]]]
```

Így jelenik meg egy, a csatolási útvonalnévvel ellátott adatbázis-objektumra történő hivatkozás (adatbázis-objektum útvonal). Az adatbázis-objektumneveket ponttal kell elválasztani egymástól. Az opcionális csatolási útvonalnév csak a bejegyzett táblázat-hivatkozásra vonatkozóan lehet megadva. Ha az adatbázis-objektumra történő hivatkozás egy üres karaktersorozat, vagy nincs megadva, a rendszer úgy tekinti, mintha az minden környezetre és azok csatolásaira vonatkozna. Egy üres adatbázis-objektumra történő hivatkozás felhasználható minden csatolás végrehajtására vagy az összes csatolási útvonalnév listájának megszerzésére. Ezen kívül használható az **ase_dolinst** függvényben szereplő környezetek listájának megszerzésére is.

Megjegyzés A pont szimbólum nem használható az adatbázis-objektumnév helyett, ha a DBMS támogatja a funkciót. Ebben az esetben ajánlott a csillag karakter használata a kihagyott adatbázis-objektum helyett. Például ha a DBMS nem támogatja a katalógus lehetőséget, a táblázathoz elérési útvonala a következő lehet: ASETUT.*.SCH.ASETUT.

Oszlopnév—*column_name*

```
behatárolt_azonosító|szabályos_azonosító
```

Az ASE program a táblázat-oszlopneveket behatárolt és szabályos azonosítókként támogatja. Lásd az *AutoCAD Developers Guide* „Syntax of SQL2 Implementation" c.,C függelékét.

Kulcsérték—*key*

(*key_val1 key_val2 ...*)

Ez a kulcsérték határozza meg, a bejegyzett adatbázis-táblázat adott sorát. A kulcsoszlop-értékek sorozata és mennyisége meg kell hogy feleljen a megadott csatolási útvonal kulcsoszlopainak. A kulcsértékek formátumának az alábbi elrendezésben kell állnia:

AutoLISP adattípus	*ASI adattípus*
Valós	REAL, FLOAT, DOUBLE PRECISION, NUMERIC, DECIMAL, INTEGER, SMALLINT
Karakterlánc	CHARACTER, DATE, TIME, TIMESTAMP, INTERVAL

A BIT, VÁLTOZÓ BIT és Megvalósítás-által definiált adattípus-értékek nem használhatóak kulcsértékekként. Ezen értékek a csatoló DA Oszlopérték attributumában használhatók, és üres karaktersorozatoknak kell lenniük.

Csatolás azonosító—*link_id*

Ez a hivatkozik a csatolás azonosítók rendezett halmazára. A csatolás-azonosító valós számként jelenik meg, ami egy szerkesztési alkalom során ugyanaz az érték.

Csatolás kiválasztás—*link_sel*

Ez a hivatkozás vonatkozik a csatolás-rendezésre, ami a csatolás-azonosítók sorozatát tartalmazza. A csatolás kiválasztás valós számként jelenik meg, ami egyedülálló egy rajzon belül. A csatolás kiválasztás nem tartalmazhat duplázott csatolási azonosítókat. A már rendelkezésre álló azonosítóknak a csatolás kiválasztáshoz való hozzáadását a rendszer figyelmen kívül hagyja. A valós szám a csatolás azonosító pozícióját a csatolási sorozatban határozza meg.

Végrehajtás

Minden API függvény az ASE végrehajtást használja, ami tevékenységek sorozata egy egészként végrehajtva. Ezen tevékenységek eredményeit a rendszer vagy elfogadja, vagy törli. Ezek a műveletek változásokat eredményezhetnek rajzban, csatolásokban és adatbázis adatokban. Ilyen esetben az ASE párbeszédfelületben lévőkkel megegyező szabályok érvényesek. Valahányszor egy AutoLISP felületfüggvény elér egy adatbázisbeli adatot, ez egy, a DBMS felé irányuló műveletként valósul meg. Amennyiben nincs hiba a függvény végrehajtása során, a végrehajtás lezárul egy akció

művelettel. Hiba fellépése esetén mindegyik, adatbázisbeli adatokat elérő függvény lezárja a pillanatnyi végrehajtást egy vissza művelettel. Végrehajtási problémák léphetnek fel, amikor az AutoLISP az ASILISP programot használja adatbázisbeli adatok elérésére. Szintén előfordulhatnak konfliktusok, ha az alkalmazás meghív néhány ASE Program Összekötő függvényt vagy ASE parancsot egy befejezetlen végrehajtás ideje alatt úgy, hogy ez a művelet és az ASE művelet ugyanarra az adatbázisbeli adatra vonatkozik. Ajánlott minden ASILISP végrehajtás megszakítása az ASE program összekötő függvények, vagy az adatbázisbeli lezárt adatokat elérni képes ASE parancsok hívása előtt.

Általánosító megjegyzések

Az ASE program 13. verziója a neveket és a karakter értékeket a rajzi ASCII karakterkészletben tárolja. Az API függvények ezek 8 bites karakterekkénti megvalósítását támogatják az alapértelmezett karakterkészletben, az *asi.ini* fájlban megadva. Néhány összekötő függvény szintén az alapértelmezett karakterkészlet szerint ad vissza neveket és karakteradatokat.

Hibadiagnosztika

Minden ASE parancs és felületfüggvény létrehozhat hibát. Az ASE program a legutolsó hibaadatot az átfogó diagnosztikai területen tartja. Ez a terület a hibaleírók sorozatát tartalmazza. Minden leírót az alábbi adatok minősítenek:

Hibaleíró-kód A kód a hibahelyzet típusát minősíti.

Jellemző hibakód A helyzetre jellemző kód

Jellemző hibaüzenet A helyzetre jellemző üzenet

Minden egyes hibaleíróra vonatkozó hibaleíró szám, hibaleíró kód, jellemző hibakód és hibaüzenet megkapható hibadiagnosztikai függvények segítségével. Bármely API függvény vagy parancs végrehajtása előtt kitörli az átfogó diagnosztikai területet.

9

KIVÁLASZTÁSI HALMAZOKAT, OBJEKTUMOKAT ÉS SZIMBÓLUMTÁBLÁKAT KEZELŐ FÜGGVÉNYEK

A kiválasztási halmazokat és objektumokat kezelő függvények többsége *rajzelemnév* szerint azonosítja a halmazt vagy rajzelemet. Mielőtt egy AutoLISP alkalmazás műveletet végezhetne egy kiválasztási halmazzal vagy objektummal, le kell kérdeznie annak aktuális nevét valamely erre alkalmas függvény segítségével.

A kiválasztási halmazok esetében, amelyek csak az adott szerkesztési alkalommal léteznek, a nevek mulandósága nem okoz problémát; nem így az objektumok esetében, amelyeket az AutoCAD elment a rajz adatbázisában. Ha egy alkalmazásnak ugyanabban a rajzban (rajzokban) különböző alkalmakkor is ugyanarra a rajzelemre kell hivatkoznia, akkor erre a célra a rajzelemek *sorszámát* használhatja.

Az AutoLISP szimbólumtáblákat használ a rajzhoz kapcsolódó grafikus és nem grafikus adatok (például fóliák, vonaltípusok és blokk-definíciók) listáinak kezelésére. Minden szimbólumtábla-bejegyzésnek van egy kapcsolódó rajzelem neve és sorszáma, amely más AutoCAD rajzelemekhez hasonlóan kezelhető.

Kiválasztási halmazok kezelése

Az AutoLISP programban megtalálhatók a kiválasztási halmazok kezelésére szolgáló függvények. A kiválasztási halmaz kezelésére szolgáló függvények az alábbiak:

ssadd	ssget	ssmemb
ssdel	sslength	ssname

Az **ssget** függvény használata a legáltalánosabb módszer a kiválasztási halmazok létrehozására. Az alábbi módszerek egyikével tud létrehozni egy kiválasztási halmazt:

- Az objektumok explicit meghatározásával az Utolsó, Előző, Ablak, Metsz, APoligon, Keresztező, MPoligon vagy Felfűz opciókkal.
- Egy különálló pont megadásával.
- Az egész adatbázis kiválasztásával.
- Objektumok kiválasztását kérve a felhasználótól.

A *szűrést* bármelyik opcióval együtt használhatja azon tulajdonságok és feltételek listájának megadására, amelyeknek a kiválasztott objektumoknak meg kell felelniük.

Megjegyzés A kiválasztási halmaz és rajzelemnevek átmenetiek, azok csak az aktuális szerkesztési alkalomra vonatkoznak.

Az **ssget** függvény első argumentuma a kiválasztási opciót meghatározó karakterlánc. A következő két argumentum a *pont1* és *pont2*, amelyek az opciónak megfelelő pontokat adják meg (ezeket el kell hagyni, ha az adott opció esetében értelmetlenek). A sokszögek révén történő kiválasztást lehetővé tevő módszerek (Felfűz, Metsző sokszög és Ablaksokszög) esetén egy *pontlista* argumentum megadása szükséges. Az utolsó, a *szűrőlista* argumentum opcionális. Ha a *szűrőlista* szerepel a függvényhívásban, akkor az a szűrésre felhasznált rajzelem-adatmezők értékének listáját határozza meg. A szűrők részletesebb ismertetését a "Kiválasztási halmazok szűrőlistái" című résznél, a 266. oldalon találja. A következő táblázat a lehetséges kiválasztási módok értékét és az azokkal együtt használható argumentumokat foglalja össze (további argumentumként valamennyi felsorolt kiválasztási módszer esetén szerepelhet egy *szűrőlista)*:

A ssget függvénykiválasztási opciói

Mód	*Kiválasztási módszer*	*Prototípusok*
semmi	A felhasználó által, vagy egyetlen ponttal történő kiválasztás (ha *pont1* adott)	**(ssget)** vagy **(ssget pont1)**
"U"	A képernyőn látható utoljára létrehozott rajzelem	**(ssget "U")**
"E"	Előző kiválasztott kiválasztási halmaz	**(ssget "E")**
"V"	előVálasztott kiválasztási halmaz (a PICKFIRST mód Bekapcsolt állapotában létrehozott előző elemsorozat)	**(ssget "V")**
"A"	Ablakkal történő kiválasztás	**(ssget "A" pont1 pont2)**
"M"	Metsző ablakkal történő kiválasztás	**(ssget "M" pont1 pont2)**
"F"	Felfűzéssel (nyitott poligonnal) történő kiválasztás	**(ssget "F" pontlista)**
"MP"	Metsző Poligonnal történő kiválasztás	**(ssget "MP" pontlist)**
"AP"	Ablak Poligonnal történő kiválasztás	**(ssget "AP" pontlist)**
"X"	A rajz összes rajzelemének kiválasztása	**(ssget "X")**

Megjegyzés Amennyiben az "X" *mód*-ot *szűrőlista* nélkül adja meg, akkor az **ssget** függvény az adatbázisban található *összes* rajzelemet kiválasztja, köztük azokat is, amelyek kikapcsolt vagy befagyasztott fóliákon helyezkednek el, vagy éppen nem láthatók a képernyőn.

A következő programozási példák az **ssget** függvény egy-egy jellegzetes hívását mutatják be:

```
(setq pt1 '(0.0 0.0 0.0)
      pt2 '(5.0 5.0 0.0)
      pt3 '(4.0 1.0 0.0)
      pt4 '(2.0 6.0 0.0)
)
```
Beállítja a pt1, pt2, pt3 és pt4 pontok értékét.

```
(setq ss1 (ssget))
```
A felhasználótól általános objektumkiválasztást kér, és a kapott elemeket egy kiválasztási halmazban helyelyezi el.

```
(setq ss1 (ssget "E"))
```
Kiválasztási halmazt képez a legutoljára kiválasztott objektumokból.

```
(setq ss1 (ssget "U"))
```
Kiválasztási halmazt képez a képernyőn láthatók közül az adatbázishoz utoljára hozzáadott objektumból.

```
(setq ss1 (ssget pt2))
```
Kiválasztási halmazt képez az (5,5) ponton átmenő objektumból.

```
(setq ss1 (ssget "A" pt1 pt2))
```
Kiválasztási halmazt képez a (0,0) és az (5,5) pontok által kifeszített ablak belsejében levő objektumokból.

```
(setq ss1 (ssget "F" (list pt2 pt3 pt4)))
```
Kiválasztási halmazt képez az (5,5), (4,1) és a (2,6) pontok által meghatározott nyitott poligont metsző objektumokból.

```
(setq ss1 (ssget "AP" (list pt1 pt2 pt3)))
```
Kiválasztási halmazt képez a (0,0), (5,5) és a (4,1) pontok által meghatározott poligonon belül levő objektumokból.

```
(setq ss1  (ssget "X"))
```
Kiválasztási halmazt képez az adatbázisban található összes objektumból.

Amikor egy alkalmazásnak már nincs szüksége egy kiválasztási halmazra, fontos, hogy törölje azt a memóriából. Ezt úgy teheti meg, hogy `nil` értéket rendel hozzá.

```
(setq ss1 nil)
```

Megjegyzés Nagyszámú kiválasztási halmaz egyidejű használata nem ajánlott. Egy AutoLISP alkalmazásban egyszerre nem lehet 128-nál több nyitott kiválasztási halmaz. Elképzelhető, hogy az Ön rendszere esetében még valamivel kisebb is ez a szám. Amikor ezt a határt eléri, az AutoCAD megtagadja további kiválasztási halmazok létrehozását. A megnyitott halmazok számát érdemes inkább a legalacsonyabb értéken tartani, és a már felesleges halmazoknak, amint lehet, `nil` értéket adni. A kiválasztási halmazok maximális számának elérésekor meg kell hívnia a **gc** függvényt (a hulladékgyűjtő folyamatra vonatkozó tudnivalókat tekintse meg a "Csomóponti terület" című résznél, az 564. oldalon) ahhoz, hogy a további **ssget** hívások működjenek.

Kiválasztási halmazok szűrőlistái

A kiválasztási halmaz szűrőlistája egy asszociációs lista, amely DXF csoportkódokat használ (tekintse meg a 16., "Alkalmazás-specifikus DXF csoportkódok" című fejezetét, a csoportkódok listájának

leírására), ugyanabban a formátumban, mint amilyet az **entget** függvény szolgáltat. Az **ssget** függvény valamennyi csoportkódot elfogadja, kivéve a rajzelem-neveket (–1-es csoport), sorszámokat (5-ös csoport) és a bővített adatokat (nagyobb, mint 1000). A *szűrőlista* érvénytelen csoportkódjait az **ssget** függvény figyelmen kívül hagyja. A bővített adatokkal rendelkező objektumok keresése a –3-as kóddal történhet, mint azt a "Bővített adatok szűrése" című résznél, a 269. oldalon ismertetjük.

Ha az **ssget** hívásban utolsó argumentumként egy *szűrőlista* is adott, akkor a függvény végigpásztázza a kiválasztott objektumokat, és egy olyan kiválasztási halmazt képez, amely a megadott követelményeknek eleget tevő összes fő rajzelem nevét tartalmazza. Ezzel a módszerrel létrehozhat például egy olyan kiválasztási halmazt, amely tartalmazza az adott típusú, adott fólián található illetve adott színű rajzelemeket.

A *szűrőlista* határozza meg a rajzelemek vizsgálni kívánt tulajdonságát (tulajdonságait), és a feltételt kielégítő értékeket.

Az alábbi négy példa a *szűrőlista* használatának módszereit szemlélteti a *mód* argumentumnak megfelelő különböző kiválasztási opciók esetén.

```
(setq ss1 (ssget
  '((0 . "TEXT")))
)
```

A Felhasználótól általános objektum kiválasztást kér, de csak a szöveg-objektumokat veszi fel a kiválasztási halmazba.

```
(setq ss1 (ssget "E"
'((0 . "LINE")))
)
```

Kiválasztási halmazt képez a legutóbb kiválasztott objektumok között található vonal-objektumokból.

```
(setq ss1 (ssget "A" pt1 pt2
'((8 . "FLOOR9")))
)
```

Kiválasztási halmazt képez az ablakon belül levő, és egyben a FLOOR9 fólián található objektumokból.

```
(setq ss1 (ssget "X"
'((0 . "CIRCLE")))
)
```

Kiválasztási halmazt képez az adatbázisban található összes kör-objektumból.

Amennyiben a csoportkód és a kívánt érték is ismert, a lista megadható az imént bemutatott, aposztrofált módon. Ha

bármelyikük változóként szerepel, akkor a listát össze kell állítani (a **list** vagy a **cons** függvény segítségével).

```
(setq fól_név "3_EMELET")
(setq ss1
  (ssget "X"                          Kiválasztási halmazt képez az
    (list (cons 8 fól_név))           adatbázisban található összes olyan
                                      rajzelemből, amely a 3_EMELET fólián
                                      helyezkedik el.
  )
)
```

Ha a *szűrőlista* egynél több tulajdonságot határoz meg, akkor csak azok az objektumok kerülhetnek bele a kiválasztási halmazba, amelyek valamennyi megadott feltételnek megfelelnek. Például (folytatva az előző példát):

```
(ssget "X"  (list  (cons 0 "CIRCLE")(cons 8 fól_név)(cons 62 1)))
```

Ez a függvényhívás csak a 3_EMELET fólián lévő, vörös színű Kör-objektumokat választja ki. Az ilyen típusú feltételvizsgálat logikai ÉS műveletnek felel meg. Az objektumok tulajdonságai szerinti további vizsgálatokról a "Szűrővizsgálatok logikai csoportosítása" című résznél, a 270. oldalon lesz szó.

Az **ssget** függvény úgy végzi el egy rajz szűrését, hogy végigpásztázza a kiválasztási halmazokat, és minden fő rajzelem mezőt összehasonlít a megadott szűrőlistával. Egy rajzelem akkor kerül bele az eredményül kapott kiválasztási halmazba, ha tulajdonságai a szűrőlistában megadott *valamennyi* mezőnek megfelelnek; máskülönben kimarad belőle. Amennyiben a kiválasztási halmazok között nincs a megadott szűrőfeltételnek megfelelő halmaz, az **ssget** függvény `nil` értékkel tér vissza.

Megjegyzés Bizonyos csoportkódok jelentése eltérhet az egyes rajzelemek esetében, továbbá nem minden csoportkód fordul elő az összes rajzelemben. Azok a rajzelemek, amelyek nem tartalmazzák a szűrőben megadott csoportkódot, kimaradnak az **ssget** által visszaadott kiválasztási halmazból.

Amikor az **ssget** függvény szűrést végez a rajzon, az eredményül kapott kiválasztási halmaz mind papír-, mind modelltérbeli elemeket tartalmazhat. Amikor azonban a kiválasztási halmazt átadja egy AutoCAD parancsnak, az csak az aktuális térben lévő rajzelemeket fogja figyelembe venni. (A rajzelem 67-es csoportjának értéke határozza meg, hogy melyik térhez tartozik. Erről a 16., "Alkalmazás-specifikus DXF csoportkódok" című fejezetben olvashat).

Globális karakterminták a szűrőlistákban

A szűrőlistákban megadott szimbólumnevek globális karaktereket is tartalmazhatnak. Az **ssget** függvény által elfogadott globális karakterminták azonosak a **wcmatch** függvény esetében

megengedettekkel. Ismertetésük az "Összehasonlítás globális karakterekkel" című résznél, a 204. oldalon , valamint a **wcmatch** függvény leírásánál, a 13. fejezetben található.

Megjegyzés Névtelen blokkokra vonatkozó szűrőlistákban a * karakter elé vezérlőkarakterként a fordított aposztróf (`) jelet kell beilleszteni, máskülönben a * jelet az **ssget** függvény globális karakterként értelmezi. Például egy *U2 nevű névtelen blokkot a következő kifejezéssel választhat ki:

```
(ssget "X" '((2 . "`*U2")))
```

Bővített adatok szűrése

Az **ssget** függvény *szűrőlista* argumentumának felhasználásával kiválaszthatja az összes olyan bővített rajzelemadatot, amely egy adott alkalmazáshoz tartozik (további tudnivalók a "Bővített adatok—Xdata" című résznél, a 289. oldalon) Ez a –3-as csoportkód segítségével történhet, például így:

```
(ssget "X" '((0 . "CIRCLE") (-3 ("ALKNÉV"))))
```

Ez a függvényhívás minden olyan kört kiválaszt, amely az "ALKNÉV" alkalmazáshoz tartozó bővített adattal rendelkezik. Amennyiben a –3-as csoport listája egynél több alkalmazás nevét tartalmazza, akkor azok ÉS kapcsolatban állnak egymással, tehát a rajzelemnek rendelkeznie kell az összes megadott alkalmazáshoz tartozó bővített adatokkal. Ily módon a következő függvényhívás minden olyan kört kiválaszt, amely egyaránt rendelkezik az "ALK1" és az "ALK2" alkalmazáshoz tartozó bővített adatokkal.

```
(ssget "X" '((0 . "CIRCLE") (-3 ("ALK1")("ALK2"))))
```

A globális karakterminták használata megengedett, ezért akár az első vagy a második, akár mindkét függvényhívás kiválaszt minden olyan kört, amely akár egyik, akár mindkét alkalmazáshoz tartozó bővített adattal rendelkezik.

```
(ssget "X" '((0 . "CIRCLE") (-3 ("ALK[12]"))))
(ssget "X" '((0 . "CIRCLE") (-3 ("ALK1,APP2"))))
```

Relációvizsgálat

Egyéb rendelkezés hiányában a *szűrőlista* elemeit a függvény "egyenlőségre" vizsgálja. A számszerű értékeket tartalmazó csoportok (egészek, valós számok, pontok és vektorok) esetében egyéb relációk is megadhatók egy speciális –4-es csoportkód beiktatásával, amely egy relációjelet határoz meg. A –4-es csoport értéke egy karakterlánc, amely a *szűrőlista* következő csoportjának vizsgálatára vonatkozó relációjelet tartalmazza. A további tudnivalókat lásd a "Relációvizsgálat" című résznél, az 501. oldalon.

A következő függvényhívás minden olyan kört kiválaszt, amelynek sugara (csoportkódja 40) nagyobb vagy egyenlő, mint 2.0 egység:

```
(ssget "X" '((0 . "CIRCLE") (-4 . ">=") (40 . 2.0)))
```

Szűrővizsgálatok logikai csoportosítása

A "Szűrővizsgálatok logikai csoportosítása" című résznél, az 503. oldalon ismertetett relációjeleken kívül beágyazott logikai (Boole) kifejezéseket is létrehozhat a csoportok vizsgálatára, csoportosító operátorok alkalmazásával.

A csoportosító operátorokat –4-es csoportkód segítségével adhatja meg, hasonlóan a relációjelekhez. Párosával fordulnak elő, és a párosításnak a szűrőlistában tökéletesen kiegyenlítettnek kell lennie, máskülönben az **ssget** függvény hívása hibához vezet. A következő példa a csoportosító operátorok használatát mutatja be egy szűrőlistában:

```
(ssget "X"

 '(
    (-4 . "<OR")
      (-4 . "<AND")
        (0 . "CIRCLE")
        (40 . 1.0)
      (-4 . "AND>")
      (-4 . "<AND")
        (0 . "LINE")
        (8 . "ABC")
      (-4 . "AND>")
    (-4 . "OR>")
  )
)
```

Ez a függvényhívás kiválasztja mindazokat a Köröket, amelyek sugara 1.0 egység, valamint mindazokat a Vonalakat, amelyek az "ABC" fólián helyezkednek el. A csoportosító operátorok nagy- és kisbetűs írásmódja között nincs különbség.

A csoportosító operátorok a –3-as csoporton belül nem használhatók. A –3-as csoportban megadott több alkalmazásnév között ÉS kapcsolat áll fenn. Amennyiben a bővített adatok vizsgálatához csoportosító operátorokat kíván használni, akkor különálló –3-as csoportokat adhat meg, tetszés szerinti csoportosításban. Azokat a Köröket, amelyek vagy az "ALK1", vagy az "ALK2" alkalmazáshoz — de nem mindkettőhöz — tartozó bővített adattal rendelkeznek, a következőképpen választhatja ki:

```
(ssget "X"
 '((0 . "CIRCLE")
    (-4 . "<XOR")
      (-3 ("APP1"))
      (-3 ("APP2"))
```

```
        (-4 . "XOR>")
    )
)
```

Egyszerűsítheti a programírást, ha a gyakran használt csoportosító operátorokat egy-egy szimbólumhoz rendeli hozzá. Az előző példa az alábbi módon írható át (figyelje meg, hogy ebben az esetben minden listát külön aposztróffal kell bevezetni):

```
(setq <xor '(-4 . "<XOR")
      xor> '(-4 . "XOR>") )
(ssget "X"
  (list
   '(0 . "CIRCLE")
    <xor
      '(-3 ("APP1"))
      '(-3 ("APP2"))
    xor>
  )
)
```

Mint láthatja, a módszer az előnyei nem érzékelhetők; ilyen rövid programrészlet esetében nagyobb alkalmazásokban azonban hasznos lehet.

Műveletek kiválasztási halmazokkal

Egy már létező kiválasztási halmazhoz további rajzelemeket adhat, vagy ki is vonhat rajzelemeket belőle a **dd** illetve **ssdel** függvények segítségével. Amint az alábbi példában látható, az **ssadd** függvény új kiválasztási halmaz létrehozására is használható. A következő programrészlet egy olyan kiválasztási halmazt hoz létre, amely az aktuális rajz első és utolsó rajzelemét foglalja magába (az **entnext** és **entlast** függvények ismertetése később következik ebben a fejezetben).

```
(setq elsnév (entnext))                  a rajz első elemét keresi ki
(setq utonév (entlast))                  a rajz utolsó elemét keresi ki
(if (not elsnév)
  (princ "\nA rajzban egy rajzelem sincs. ")
  (progn
    (setq ourset (ssadd elsnév))         kiválasztási halmazt képez az első
                                         rajzelemből
    (ssadd utonév ourset)                az utolsó rajzelemet hozzáadja
                                         ugyanahhoz a kiválasztási halmazhoz
  )
)
```

Ez a példaprogram akkor is hibátlanul működik, ha az adatbázisban csak egyetlen rajzelem található (ilyen esetben az **entnext** és az **entlast** ugyanannak a rajzelemnek a nevével tér vissza). Ha az **ssadd** függvénynek egy olyan rajzelem nevét adja meg, amely már szerepel a kiválasztási halmazban, akkor azt a függvény nem adja a halmazhoz,

de hibát *nem* jelez. A következő függvényhívás eltávolítja az első rajzelemet az előbbi példában létrehozott kiválasztási halamazból:

```
(ssdel elsnév ourset)
```

Ha a rajz több rajzelemet tartalmaz (tehát `elsnév` és `utonév` nem azonos), akkor az `ourset` halmazban csak az `utonév`, a rajz utolsó eleme található.

Az **`sslength`** függvény a kiválasztási halmazban található rajzelemek számával tér vissza, az **`ssmemb`** függvény pedig azt vizsgálja, hogy egy adott rajzelem szerepel-e a kiválasztási halmazban. Végezetül, az **`ssname`** függvény a kiválasztási halmaz egy adott indexű elemének nevével tér vissza (a halmaz elemeinek számozása nullától kezdődik). A következő programrészlet az **`ssname`** függvény néhány hívását mutatja be:

```
(setq khal (ssget))                    kiválasztási halmazt képez (a felhasználó
                                       közreműködésével)
(setq elem1 (ssname khal 0))           kiolvassa a khal első elemének nevét
(setq elem4 (ssname khal 3))           kiolvassa a khal negyedik elemének nevét
(if (not elem4)
  (princ "\nLegalább négy rajzelem kiválasztása szükséges. ")
)
(setq iut (sslength khal))             megkeresi a khal utolsó elemének indexét
                                       kiolvassa a khal utolsó elemének nevét
(setq elemu (ssname khal (1- iut)))
```

Megjegyzés A kiválasztási halmazok sohasem tartalmazzák többszörösen ugyanazt a rajzelemet, tekintet nélkül arra, hogy milyen módon adódott hozzá a halmazhoz. Ha egy rajzelemet egynél többször próbál hozzáadni a kiválasztási halmazhoz, a kísérlet egyszerűen eredménytelen lesz. Ennek következtében az **`sslength`** függvény mindig pontosan az adott kiválasztási halmazban található rajzelemek számával tér vissza.

Objektumkezelés

Az AutoLISP objektumkezelő függvényeket is tartalmaz. Ezek a következők:

entdel	entlast	entmod	entupd
entget	entmake	entnext	handent

Az objektumkezelő függvények két csoportba oszthatók: vagy visszakeresik egy részleges objektum rajzelemnevét, vagy visszakeresik, illetve módosítják, a rajzelemadatokat.

Rajzelemnév függvények

Ahhoz, hogy egy AutoLISP alkalmazás műveletet végezhessen egy rajzelemmel, le kell kérdeznie annak nevét. A nevet később felhasználhatja a rajzelem-adatokat vagy kiválasztási halmazokat kezelő függvények hívásakor. Az ebben a részben ismertetett függvények közül az **entsel** és a **entsel** a rajzelem nevén kívül az alkalmazás számára hasznos további információkat is szolgáltat.

Mindkét függvény egy rajzelem kiválasztását várja a felhasználótól interaktív módon, egy pont kijelölésével a grafikus képernyőn. Az elemneveket kezelő összes többi függvény akkor is képes elérni egy rajzelemet, ha az nem látható a képernyőn, vagy egy befagyasztott fólián helyezkedik el. Az **entsel** függvény arra kéri a felhasználót, hogy egy pontnak a grafikus képernyőn történő kijelölésével válasszon ki egy rajzelemet. Az **entsel** mind a kiválasztott rajzelem nevét, mind a kijelölt pontot visszaadja. A rajzelemekkel végezhető egyes műveletekhez szükség van annak a pontnak az ismeretére, amely által a rajzelem kiválasztásra került. Példa erre egy sor AutoCAD parancs, köztük az ELTÖR, a METSZ és az ELÉR. A **nentsel** függvény részletes ismertetésére az "Rajzelemek összefüggései és koordináta-transzformációs adatok" című résznél, a 275. oldalon kerül sor. Ezek a függvények az **initget** függvény hívásában előzőleg definiált kulcsszavakat is figyelembe veszik.

Az **entnext** függvény az egymás után következő rajzelemek nevét olvassa ki. Amennyiben az **entnext** hívásában nem szerepel argumentum, a függvény az adatbázisban található legelső rajzelem nevével tér vissza. Amennyiben a függvény argumentuma az adatbázisban szereplő egyik rajzelem neve, az **entnext** az azt követő rajzelem nevével tér vissza.

Ez a programrészlet azt mutatja be, hogyan használható az **ssadd** függvény az **entnext** függvénnyel együtt kiválasztási halmazok létrehozására, és arra, hogy egy meglevő halmazhoz további elemeket adjon.

```
(setq e1  (entnext))
(if  (not e1)                          e1 értéke az első rajzelem neve lesz
  (princ "\nA rajzban egy rajzelem sincs. ")
  (progn
    (setq ss  (ssadd))                 létrehozza az es nevű üres kiválasztási
                                       halmazt
    (ssadd e1 ss)                      hozzáadja az e1 elemet az es kiválasztási
                                       halmazhoz
    (setq e2 (entnext e1))             kiolvassa az e1 utáni rajzelem nevét
    (ssadd e2 ss)                      hozzáadja az e2 elemet az es kiválasztási
                                       halmazhoz
  )
)
```

Az **entlast** függvény az adatbázisban található utolsó rajzelem nevét olvassa ki. Ez a legutóbb létrehozott fő rajzelem, így az **entlast** függvény hívásával megtudhatja annak a rajzelemnek a nevét, melyet előzőleg a **command** függvény segítségével hozott létre.

Az **entnext** függvény által visszaadott elemnevet ugyanabban a változóban helyezheti el, mint amellyel a függvényt meghívta. Így lényegében egyetlen elemnév-változó felhasználásával "végiglépkedhet" az egész adatbázison. Például:

```
(setq egy_elem (entnext))   kiolvassa az első rajzelem nevét

 (while egy_elem
  .
  .                                  következő rajzelem feldolgozása
  .
   (setq egy_elem (entnext egy_elem))
)                                    egy_elem értéke most nil
```

Rajzelem-sorszámok és használatuk

A **handent** függvény egy megadott sorszámmal rendelkező rajzelem nevét olvassa ki. A rajzelemek nevéhez hasonlóan, a sorszámok szintén egyediek egy rajzban. Azonban az elemnevektől *eltérően*, a sorszám mindaddig állandó marad, amíg a rajzelem létezik. Egy adott adatbázisban műveleteket végző AutoLISP alkalmazás a **handent** függvény segítségével kapja meg egy használni kívánt rajzelem aktuális nevét. Egy kiválasztott objektum sorszámát a DDMÓDOSÍT parancs segítségével kapja meg.

A következő programrészlet a **handent** függvény segítségével kiolvassa és megjeleníti egy rajzelem nevét:

```
(if  (not (setq e1  (handent "5a2")))
  (princ "\nNem létezik ilyen sorszámú rajzelem. ")
  (princ e1)
)
```

Egy adott szerkesztési alkalommal ez a kód jelenhet meg:

<Elem neve: 60004722>

Ugyanennek a rajznak egy másik alkalommal történő szerkesztésekor egy ettől teljesen eltérő szám jelenhet meg. A kód azonban mindkét esetben ugyanarra a rajzelemre vonatkozik.

A **handent** függvény egy további célra is felhasználható. Az adatbázisból törölt (az **entdel** függvény segítségével, melynek ismertetésére a következő részben kerül sor) rajzelemek nincsenek véglegesen eltávolítva mindaddig, amíg az aktuális rajz szerkesztése be nem fejeződik. Ez azt jelenti, hogy a törölt rajzelemek neve is hozzáférhető a **handent** függvény segítségével, így azok egy újabb **entdel** hívással visszaállíthatók a rajzban.

Megjegyzés A blokkdefiníciók, beleértve a blokkok alárendelt rajzelemeit is, sorszámot kapnak.

Az XREF Illeszt paranccsal beillesztett rajzok elemei ténylegesen nem képezik részét az aktuális rajznak: sorszámuk változatlan, a **handent** függvénnyel azonban nem elérhetők. Ha viszont a rajzokat a BEILL, BEILL *, XREF Csatol (XCSATOL) vagy részleges DXFBE parancsok révén fűzi egybe, akkor a beillesztett rajz elemei elveszítik sorszámukat (feltéve, hogy volt az eredeti rajzban), és új sorszámértéket kapnak, biztosítva ezáltal a sorszámok egyedi jellegének megőrzését az aktuális rajzban.

Rajzelemek összefüggései és koordináta-transzformációs adatok

A **nentsel** és a **nentselp** függvények az **entsel** függvényhez hasonlók, azzal a különbséggel, hogy két további értéket is visszaadnak, amelyek blokkhivatkozásokon belüli beágyazott elemek kezelését segítik elő.

További különbség e függvények között, hogy amikor a felhasználó egy **nentsel** függvényhívásra adott válaszként egy összetett rajzelemet jelöl ki, illetve amikor a **nentselp** egy összetett rajzelemet választ ki, akkor ezek a függvények a kiválasztott alárendelt rajzelem nevével, és *nem* az összetett rajzelem fejlécének nevével térnek vissza, ahogyan azt az **entsel** függvény teszi. Például, amennyiben a felhasználó egy vonalláncot jelöl ki, a **nentsel** függvény egy csomópont elemmel tér vissza a vonallánc fejléce helyett. A vonallánc fejlécének megkereséséhez az alkalmazásnak végig kell haladnia az alárendelt rajzelemeken egészen a sorvég (Seqend) elemig, és annak −2-es csoportjából kell kiolvasnia a fejléc nevét. Ugyanez érvényes akkor is, ha a felhasználó attribútumokat választ ki egy beágyazott blokkhivatkozásban. A **nentselp** függvény általában előnyösebb, mint a **nentsel**, mivel egy ugyanolyan formátumú transzformációs mátrixot ad vissza, amilyet a **grvecs** függvény vár.

A **nentsel** függvény által visszaadott speciális tételek közül az első a Modell—Világ transzformációs mátrix. Ez egy olyan lista, amely négy — koordinátacsoportokat tartalmazó — allistából áll. Ennek a mátrixnak a segítségével lehet egy rajzelem definíciós adatait egy belső Modell koordináta-rendszerből (MKR) a Világ koordináta-rendszerbe (VKR) transzformálni. Az MKR origóját a kiválasztott elemet tartalmazó blokk (ez vonatkozik külső referenciákra is) beillesztési pontja határozza meg. Tengelyeinek iránya a blokk létrehozásakor használt FKR tengelyeinek irányával azonos.

A másik speciális tétel egy lista, amely a kiválasztott rajzelemet magában foglaló blokk elemnevét tartalmazza. Ha a kiválasztott rajzelem egy beágyazott blokk (blokk a blokkban) része, akkor a lista kiegészül mindazon blokkok elemnevével, amelyekbe a kiválasztott

rajzelem beágyazódott, a "legbelső" blokktól kezdve egészen a rajzba beillesztett "legkülső" blokkig.

`(<Elem neve: enév1>`	*A rajzelem neve*
`(Px Py Pz)`	*Kijelölési pont*
`( (X0 Y0 Z0)`	*Modell-Világ*
`(X1 Y1 Z1)`	*transzformációs mátrix*
`(X2 Y2 Z2)`	
`(X3 Y3 Z3)`	
`)`	
`(<Elem neve: enév2>`	*A kiválasztott objektumot tartalmazó legmélyebben beágyazott blokk neve*
`<Elem neve: enévn>)`	*A kiválasztott objektumot tartalmazó legkülső blokk neve*

A következő példa egy blokkot hoz létre a **`nentsel`** számára.

Parancs: vonal
Ponttól...: **1,1**
...Pontig: **3,1**
...Pontig: **3,3**
...Pontig: **1,3**
...Pontig: **c**
Parancs: **blokk**
Blokk név (vagy ?): **négyzet**
Beillesztés bázispontja: **2,2**
Válasszon objektumokat: *Válassza ki az imént rajzolt négy vonalat.*
Válasszon objektumokat: ↵

Ezután illessze be a Blokkot egy, a *Z* tengely körül 45 fokkal elforgatott FKR-be:

Parancs: **fkr**
Origó/ZTeng/3pont/Rajzelem/Nézet/X/Y/Z/Előző/Visszaállít/Ment/Töröl/?/
<Világ>: **z**
Forgatási szög a(z) Z tengely körül <0>: **45**
Parancs: **beill**
Blokknév (vagy ?): **négyzet**
Beillesztés pontja: **7,0**
X méretarány-tényező <1> / Sarok / XYZ: ↵
Y méretarány-tényező (alapértelmezés=X): ↵
Forgatási szög: ↵

Válassza ki a négyzet bal alsó oldalát a **`nentsel`** függvénnyel.

```
(setq nadat (nentsel))
```

Ezzel a `nadat` szimbólumhoz az alábbihoz hasonló listát rendeli hozzá:

```
(<Elem neve: 400000a0>          Rajzelem neve
 (6.46616 -1.0606 0.0)          Kijelölési pont
 ((0.707107 0.707107 0.0)       Modell-Világ
  (-0.707107 0.707107 0.0)      Transzformációs mátrix
  (0.0 -0.0 1.0)
  (4.94975 4.94975 0.0)
 )
 (<Elem neve:6000001c>)         A kiválasztott objektumot tartalmazó
                                blokk neve
)
```

Miután rendelkezésére áll a rajzelem neve és a Modell—Világ transzformációs mátrix, az MKR-ből a VKR-be tudja transzformálni az elemdefiníciós adatok pontjait. Az MKR koordinátákban kifejezett definíciós pontokat úgy kaphatja meg, ha az **`entget`** és **`assoc`** függvényeket alkalmazza a rajzelem nevére. Ezután helyettesítse be a pontok koordinátáit és — az első **`nentsel`** hívásból származó — Modell—Világ transzformációs mátrix elemeit az alábbi képletekbe.

Amennyiben a kiválasztott rajzelem nem beágyazott, akkor a transzformációs mátrix az egységmátrix lesz:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

A következő egyenletek egy pont vagy vektor transzformálását mutatják be:

$$X' = XM_{00} + YM_{10} + ZM_{20} + M_{30}$$

$$Y' = XM_{01} + YM_{11} + ZM_{21} + M_{31}$$

$$Z' = XM_{02} + YM_{12} + ZM_{22} + M_{32}$$

Az M_{ij} értékek — ahol $0 \le i,j \le 2$ — a Modell—Világ transzformációs mátrix koordinátái; X, Y és Z az elemdefiníciós adatpontot kifejező MKR koordináták; X', Y' és Z' pedig az eredményül kapott elemdefiníciós adatpontot kifejező VKR koordináták.

Megjegyzés Ha egy vektort és nem pontot kíván transzformálni, akkor az $\begin{bmatrix} M_{30} & M_{31} & M_{32} \end{bmatrix}$ eltolási vektort (a transzformációs mátrix negyedik oszlopából származik) hagyja el a műveletből.

A következő példa egy blokkdefinícióban lévő vonal MKR-beli kezdőpontjának (csoportkódja 10) meghatározását mutatja be. Az

utasítás az elemadatot (melyet a korábbi **nentsel** hívás kapott elemnév felhasználásával olvas ki) az `eadat` szimbólumban helyezi el, és a következő eredménnyel tér vissza.

(setq edata (assoc 10 (entget (car ndata))))

```
(10 -1.0 1.0 0.0)
```

A következő utasítás a Modell—Világ transzformációs mátrixot tartalmazó allistát a `matrix` szimbólumban helyezi el,

```
(setq mátrix (caddr ndata))
```

és a következő eredménnyel tér vissza:

```
((0.707107 0.707107 0.0)     X transzformáció
 (-0.707107 0.707107 0.0)    Y transzformáció
 (0.0 -0.0 1.0)              Z transzformáció
 (4.94975 4.94975 0.0)       elmozdulás a VKR-hez képest
)
```

Az *X*'-re vonatkozó transzformációs képlet alkalmazásával alakítsa át a vonal kezdőpontjának *X* koordinátáját MKR koordinátából VKR koordinátává. A `válasz` szimbólumban helyezze el az eredményt:

```
(setq válasz
  (+                                                  összeadja:
    (* (car (nth 0 mátrix))(cadr eadat))              M00 * X
    (* (car (nth 1 mátrix))(caddr eadat))             M10 * Y
    (* (car (nth 2 mátrix))(cadddr eadat))            M20 * Z
    (car (nth 3 mátrix))                              M30
  )
)
```

amelynek értéke 3.53553, ez a kiválasztott vonal kezdőpontjának VKR-beli *X* koordinátája.

Rajzelem-elérési függvények

A rajzelem-elérési függvények viszonylag lassúak. Ezért a leghelyesebb egy adott rajzelem (vagy szimbólumtábla-bejegyzés) tartalmát egyszer megszerezni és az információt a memóriában tárolni ahelyett, hogy az AutoCAD rendszertől ismételten ugyanazokat az adatokat kérné. Győződjön meg az adatok érvényességéről; ha a felhasználónak lehetősége van megváltoztatni a rajzelemet vagy a szimbólumtábla-bejegyzést, akkor újra meg kell hívnia a rajzelemet elérő függvényt az adatok érvényességének biztosítására.

Rajzelemadatokat kezelő függvények

Az ebben a részben ismertetett függvények a rajzelemadatokkal végeznek műveleteket, és segítségükkel módosítható az aktuális rajz adatbázisa.

Az **entdel** függvény törli a megadott rajzelemet. A rajzelemet az AutoCAD mindaddig nem távolítja el ténylegesen az adatbázisból, amíg az aktuális rajz szerkesztése be nem fejeződik; így ha az alkalmazás ugyanarra a rajzelemre vonatkozóan ismételten meghívja az **entdel** függvényt, akkor ez visszaállítja a rajzelemet .

Megjegyzés Az attribútumok és a vonalláncok csomópontjai nem törölhetők az őket tartalmazó rajzelemtől függetlenül, azaz az **entdel** függvény csak fő rajzelemek esetében használható. Ha egy attribútum vagy egy csomópont törlésére van szükség, akkor a **command** függvény útján az AutoCAD ATTEDIT illetve VLEDIT parancsát hívhatja segítségül.

Az **entget** függvény egy adott rajzelem definíciós adataival tér vissza egy lista formájában. A lista minden tételét egy-egy DXF csoportkód jelöli. A lista első tétele a rajzelem aktuális nevét tartalmazza.

A következő példában az aktuális rajzra az alábbi (alapértelmezés szerinti) feltételek vonatkoznak:

- A fólia 0.
- A vonaltípus CONTINUOUS (FOLYTONOS).
- A kiemelési szint 0.

A felhasználó egy vonalat rajzolt az alábbi parancssorozattal:

Parancs: **vonal**
Ponttól...: **1,2**
...Pontig: **6,6**
...Pontig: ↵

Ezt követően egy AutoLISP alkalmazás a vonal definíciós adatait az alábbi függvény segítségével kérdezi le és jeleníti meg:

```
(defun C:PRINTDXF ( )
  (setq elem (entlast))          Értéke az utolsó rajzelem neve lesz.
  (setq eleml (entget elem))     Értéke az utolsó rajzelem asszociációs listája lesz.
  (setq ct 0)                    Nullára állítja az sz1 számlálót.
  (textpage)                     Átvált a szöveges képernyőre.
  (princ "\nAz utolsó rajzelem adatai:")
  (repeat (length entl)          Annyiszor ismétli, ahány elemű a lista.
    (print (nth ct entl))        Minden tételt új sorba ír.
(setq ct (1+ ct))                Eggyel növeli a számláló értékét.
  )
  (princ)                        Csendesen lép ki
)
```

Ennek eredményeképpen a következő jelenik meg a képernyőn:

```
Az utolsó rajzelem adatai:
(-1 . <Elem neve: 60000014>)
(0 . "LINE")
(8 . "0")
(5 . "2D")
(10 1.0 2.0 0.0)
(11 6.0 6.0 0.0)
(210 0.0 0.0 1.0)
```

A lista első eleme (−1 csoportkóddal) a lista által képviselt rajzelem nevét tartalmazza. A későbbiekben ismertetésre kerülő **entmod** függvény ezt használja a módosítani kívánt rajzelem azonosítására.

A rajzelem összetevőinek kódszámai megegyeznek a DXF kódokkal, melyeknek ismertetését e kézikönyv 16., "Alkalmazás-specifikus DXF csoportkódok" című fejezet tartalmazza. Hasonló a DXF fájlformátumhoz, a rajzelem fejlécének tételei (szín, vonaltípus, vastagság, "attribútumok következnek" flag és az elemsorszám) csak akkor szerepelnek a listában, ha értékük különbözik az alapértelmezés szerinti értéktől. A DXF formátumtól eltérően, az opcionális rajzelem-definíciós mezők viszont mindenképpen szerepelnek, az alapértelmezéstől függetlenül. Ez a feldolgozás egyszerűsítését szolgálja: így az alkalmazás mindig feltételezheti, hogy ezek a mezők jelen vannak. Ugyancsak eltérően a DXF formátumtól, az összetartozó *X, Y* és *Z* koordináták egyetlen pontlistában jelennek meg, és nem mint különálló *X* (10), *Y* (20) és *Z* (30) csoportok.

Az **assoc** függvény egy adott típusú csoportot keres ki a listából. Ez a függvényhívás például az `elem1` listában található "`LINE`" (csoportkódja 0) rajzelemtípussal tér vissza.

```
(cdr (assoc 0 ent1))
```

Amennyiben a megadott DXF csoportkód nincs jelen a listában (vagy nem érvényes DXF csoportkód), akkor az **assoc** függvény `nil` értéket ad vissza.

Az **entmod** függvény módosít egy rajzelemet. Egy olyan listát továbbít az **entget** függvény által szolgáltatott formátumban, amelyben a rajzelem egyes csoportjainak értékét az alkalmazás (vélhetőleg) módosította. Ez a függvény az **entget** kiegészítője. Az AutoLISP alkalmazások által az adatbázisok felfrissítésére használt mechanizmus abból áll, hogy a felhasználó **entget** függvénnyel lekérdezi a rajzelemet, módosítja annak rajzelemlistáját, majd az **entmod** függvénnyel visszaadja a listát az adatbázisnak.

Az alábbi programrészlet kiolvassa a rajz első rajzelemének definíciós adatait, és a fóliáját ÉNFÓLIÁM-ra cseréli:

```
(setq en (entnext))                    en a rajz első elemének nevét kapja
(setq ed (entget en))                  ed az en nevű elem adatait kapja
(setq ed
  (subst  (cons 8 "ÉNFÓLIÁM")
    (assoc 8 ed)                       az ed listában a fóliacsoportot ÉNFÓLIÁM
                                       fóliára cseréli
    ed
  )
)
(entmod ed)                            módosítja az en rajzelemfóliáját az
                                       adatbázisban
```

Az adatbázisban az **entmod** függvénnyel elvégezhető módosításokra néhány megszorítás vonatkozik. Az **entmode** függvény az alábbi változtatásokat *nem képes* végrehajtani:

- Nem változtatja meg a rajzelem típusát vagy sorszámát.
- Nem változtat meg belső mezőket (a belső mezők az AutoCAD által bizonyos csoportkódokhoz rendelt értékek: −2, rajzelemnév-hivatkozás; −1, rajzelemnév; 5, elemsorszám). Minden olyan próbálkozást figyelmen kívül hagy, amely egy belső mező megváltoztatására irányul — például a fő elem nevének (csoportkódja −2) módosítása egy Seqend alárendelt rajzelemben.
- Nem változtatja meg a nézetablak-rajzelemeket. Az erre irányuló próbálkozás hibaüzenetet eredményez.

A többi megszorítás a méretezés módosítására és a sraffozásra vonatkozik.

Az AutoCAD számára valamennyi olyan objektumnak (a fóliák kivételével) felismerhetőnek kell lennie, amelyre a rajzelem listája hivatkozik. A rajzelem listájában szereplő bármely szövegstílus, vonaltípus, alak vagy blokk nevének definiáltnak kell lennie az aktuális rajzban, még *mielőtt* az **entmod** függvénynek átadja a rajzelem listáját. Egy kivétel van: az **entmod** függvény új fólianeveket is elfogad.

Ha a rajzelem listája egy előzőleg nem definiált fólia nevére hivatkozik, akkor az **entmod** függvény egy új fóliát hoz létre. Az új fólia az AutoCAD FÓLIA parancs Új opciója által használt alapértelmezés szerinti tulajdonságokat kapja.

Az **entmod** függvény *módosítani tudja* az alárendelt rajzelemeket, így a vonalláncok csomópontjait és a blokkok attribútumait.

Megjegyzés Amennyiben az **entmod** függvénnyel egy blokkdefiníción belüli rajzelemet módosít, ez annak a blokknak minden BEILL vagy XREF paranccsal beillesztett előfordulására hatással lesz. Ezenkívül, a blokkdefiníción belüli elemek nem törölhetők az **entdel** függvénnyel.

Az **entmake** függvény hívásával egy alkalmazás hozzá is adhat új rajzelemet a rajz adatbázisához. Az **entmod** függvényhez hasonlóan az **entmake** argumentuma is egy lista, az **entget** által visszaadotthoz hasonló formátumban. A lista által meghatározott új rajzelemet a függvény az adatbázis végéhez fűzi hozzá (az lesz az utolsó elem a rajzban). Ha ez egy összetett rajzelem (vonallánc vagy blokk), akkor addig nem fűzi hozzá az adatbázishoz, amíg nem teljes.

A következő programrészlet egy kört hoz létre az ÉNFÓLIÁM fólián:

```
(entmake '((0 . "CIRCLE")   objektumtípus
           (8 . "ÉNFÓLIÁM") fólia
           (10 5.0 7.0 0.0) középpont
           (40 . 1.0)       sugár
) )
```

Az **entmake** függvényre vonatkozó megszorítások hasonlók, mint az **entmod** esetében:

- A lista első vagy második tételének a rajzelem típusát kell meghatároznia. A típusnak egy érvényes DXF névnek kell lennie. Amennyiben az első tétel nem a típust határozza meg, akkor az *csak* a rajzelem nevét tartalmazó −1-es csoport lehet (az elemnév nem kerül be az adatbázisba).

- Az AutoCAD rendszernek fel kell tudnia ismerni valamennyi objektumot (a fóliák kivételével), amelyre a rajzelem listája hivatkozik. Egy kivétel van: az **entmake** függvény új fólianeveket is elfogad.

- Az **entmake** függvény figyelmen kívül hagy minden belső mezőt a kapott listában.

- Az **entmake** függvény nem hoz létre nézetablak-rajzelemet

Az **entmod** és az **entmake** függvény ugyanúgy ellenőrzi, hogy nincsenek-e ellentmondások a számukra átadott rajzelemadatokban,

mint az AutoCAD DXFBE parancsa a DXF fájlok beolvasásakor. Ha nem tudnak érvényes rajzelemet létrehozni, hibát jeleznek.

Névtelen blokkok

A blokkdefiníciók (BLOCK) táblája tartalmazhat névtelen blokkokat, melyeket az AutoCAD a sraffozási minták és az asszociatív méretezés céljára hoz létre. Az **entmake** függvényen kívül más névtelen blokkok is létrehozhatók, a `*Dnnn` (méretezések) és a `*Xnnn` (sraffozási minták) kifejezésekkel. Azokat a névtelen blokkokat, amelyekre a rajzban nincs hivatkozás, az AutoCAD a rajz betöltésekor törli BLOCK táblából. A hivatkozott (beillesztett) névtelen blokkokat *nem* távolítja el. Az **entmake** függvény segítségével létrehozhat hivatkozást (BEILL) egy névtelen blokkra (a BEILL parancsnak nem adhat át névtelen blokkot). Az **entmake** függvény a blokk átdefiniálására is használható. A blokk rajzelemei igen, de maga a blokk objektum nem módosítható az **entmod** függvénnyel.

Az AutoLISP vagy az ADS révén létrehozott névtelen blokk neve (2-es csoport) `*Unnn` alakú, ahol az *nnn* egy, az AutoCAD által generált szám. Ezenkívül a névtelen blokk *blokktípus flag*-jének (70-es csoport) alsó bitje 1-re van állítva. Amikor az **entmake** függvény egy olyan blokkot hoz létre, amelynek neve *-gal kezdődik és "névtelen" bitje be van állítva, akkor azt az AutoCAD névtelen blokknak tekinti, egy nevet rendel hozzá. Az **entmake** függvény a számára átadott névben a * után esetlegesen szereplő karaktereket figyelmen kívül hagyja.

Megjegyzés A névtelen blokknevek nem maradnak változatlanok. Bár a hivatkozott névtelen blokkok állandóak, nevük numerikus része szerkesztési alkalomról alkalomra változhat.

Összetett rajzelemek létrehozása

Az összetett rajzelemek (vonalláncok vagy blokkok) az **entmake** függvény többszöri hívásával hozhatók létre úgy, hogy mindegyik alárendelt rajzelem esetében külön meghívja azt. Amikor az **entmake** függvény először megkapja egy összetett rajzelem kezdő összetevőjét, egy ideiglenes fájlt hoz létre, amelyben összegyűjti a definíciós adatokat (és bővített adatokat, amennyiben jelen vannak; tekintse meg a "Bővített adatok—Xdata" című résznél, a 289. oldalon). Az egymást követő **entmake** hívások során a függvény minden alkalommal ellenőrzi, hogy az ideiglenes fájl létezik-e. Ha igen, akkor az új alárendelt rajzelemet hozzáfűzi a fájlhoz. Amikor az összetett rajzelem definíciója teljes (azaz, amikor az **entmake** hívásban egy megfelelő Seqend vagy Endblk elem szerepel), a függvény ellenőrzi, hogy nincs-e ellentmondás a rajzelem adatai között, és ha érvényesnek találja az elemet, akkor hozzáadja a rajz adatbázisához. A fájlt az AutoCAD akkor törli, amikor az összetett rajzelem teljesen elkészült, vagy megszakítással véget ért a létrehozása.

Megjegyzés A rajzelem addig nem jelenik meg a rajz adatbázisában, amíg az **entmake** függvény a végső sorvég (Seqend) vagy blokkvég (Endblk) elemet meg nem kapta. Az **entlast** függvény nem használható egy még nem teljes összetett rajzelem utoljára létrehozott alárendelt elemének lekérdezésére.

Az előzőekből következik, hogy az **entmake** függvénnyel egyidejűleg csak egy összetett rajzelem állítható össze. Ha egy összetett rajzelem létrehozása közben az **entmake** függvény érvénytelen adatot, vagy nem megfelelő alárendelt rajzelemet kap, akkor mind az érvénytelen rajzelemet, mind az egész összetett rajzelemet elveti. Az összetett rajzelem létrehozásának folyamata szándékosan is megszakítható az **entmake** függvény argumentumok nélküli meghívásával.

A következő példa egyetlen összetett rajzelemet, egy Vonalláncot hoz létre az **entmake** függvény ötszöri hívásával. A vonallánc vonaltípusa SZAGGAT, színe KÉK lesz. Három csomópontjának koordinátái (1,1,0), (4,6,0) és (3,2,0). A többi opcionális definíciós adat az alapértelmezés szerinti értéket veszi fel (a példa tökéletes működéséhez a SZAGGAT vonaltípust előzőleg be kell tölteni).

```
(entmake '((0 ."POLYLINE")Objektumtípus
           (62 . 5)       szín
           (6 . "szaggat") vonaltípus
           (66 . 1)       csomópontok következnek
) )
(entmake '((0 . "VERTEX")  Objektumtípus
           (10 1.0 1.0 0.0) kezdőpont
) )
(entmake '((0 . "VERTEX")  Objektumtípus
           (10 4.0 6.0 0.0) második pont
) )
(entmake '((0 . "VERTEX")  Objektumtípus
           (10 3.0 2.0 0.0) harmadik pont) )
(entmake '((0 . "SEQEND")))Sorozat vége
```

A blokkdefiníció egy blokk rajzelemmel kezdődik, és egy blokkvég (Endblk) alárendelt elemmel végződik. A blokkdefiníciók nem ágyazhatók egymásba, és nem hivatkozhatnak önmagukra. Más blokkdefiníciókra való hivatkozást azonban *tartalmazhatnak*.

Megjegyzés Mielőtt az **entmake** segítségével kíván egy blokkot létrehozni, célszerű a **tblsearch** függvény segítségével megbizonyosodni arról, hogy nincs még ilyen nevű blokk. Az **entmake** függvény nem ellenőrzi a nevek egybeesését a Blokkdefiníciók táblában, így megtörténhet, hogy átdefiniál már meglévő blokkokat.

A blokkhivatkozások tartalmazhatnak egy *"attribútumok következnek"* flag-et (csoportkódja 66). Ha ez jelen van és értéke 1, akkor az Insert rajzelem után attribútum (Attrib) rajzelemek sorozatának kell következnie. Az attribútumsort egy Seqend alárendelt rajzelem zárja le.

A vonallánc-rajzelemek mindig tartalmaznak egy *"csomópontok következnek"* flag-et (csoportkódja szintén 66). Ennek értéke mindenképpen 1, és csomópont (Vertex) rajzelemek sorozata követi, amelyet egy Seqend elem zár le.

Az összetett rajzelemek lehetnek akár a modell-, akár a papírtérben, de mindkettőben nem. Amennyiben egy összetett rajzelem felépítése közben az aktuális tér az MTÉR vagy PTÉR parancs hívásával (a **command** függvény útján) megváltozik, a következő **entmake** függvényhívás elveti az összetett rajzelemet. Ugyanez történik abban az esetben, ha egy alárendelt rajzelem 67-es csoportjának értéke nem egyezik meg a rajzelem fejrészének 67-es csoportjáéval.

Rajzelemadatokat kezelő függvények és a grafikus képernyő

Az elemadatokat kezelő függvényekkel végrehajtott rajzmódosítások a grafikus képernyőn követhetők, feltéve hogy a rajzelem törlése, visszaállítása, módosítása vagy létrehozása a képernyőn éppen látható területen, illetve fólián történt. Ez alól egy kivétel van: alárendelt rajzelem módosításakor az **entmod** függvény nem aktualizálja az egész (összetett) rajzelem képét. Ennek egyszerű a magyarázata: ha például egy alkalmazásnak egy vonallánc 100 csomópontját kellene módosítania 100 **entmod** függvényhívással, akkor elfogadhatatlanul hosszú időt venne igénybe minden egyes csomópont módosítása után újra kiszámítani és megjeleníteni a teljes vonalláncot. Ehelyett az alkalmazás elvégezheti a szükséges módosításokat az alárendelt rajzelemeken, majd egyetlen **entupd** függvényhívással az egész rajzelemet egyszerre jelenítheti meg újra.

Tételezzük fel, hogy a rajz első rajzeleme egy több csomóponttal rendelkező vonallánc. A következő program módosítja a vonallánc második csomópontját, azután regenerálja annak képét:

```
(setq e1 (entnext))                    e1 értékét a vonallánc-rajzelemnevére
                                       állítja
(setq v1 (entnext e1))                 v1 értékét az első csomópont értékére állít
                                       ja
(setq v2 (entnext v1))                 v2 értékét a második csomópont értékére
                                       állítja
(setq v2d (entget v2))                 v2 értékét a csomópontadat értékére állítja
(setq v2d
  (subst
    '(10 1.0 2.0 0.0)
     (assoc 10 v2d)                    a csomópontot az (1,2,0) pontba helyezi
                                       át
      v2d
  )
)
(entmod v2d)                           elmozdítja a csomópontot a rajzban
(entupd e1)                            Regenerálja az e1 vonallánc rajzelemet
```

Az **entupd** függvény argumentumában akár egy fő, akár egy alárendelt rajzelem megadható, az **entupd** függvény minden esetben az *egész* rajzelemet regenerálja. Bár — mint a példában látható — elsődlegesen összetett rajzelemek esetén használatos, az **entupd** függvény az aktuális rajz bármely rajzelemét képes regenerálni.

Megjegyzés Egy blokkdefiníció módosítása után az **entupd** függvényhívás nem elegendő: az AutoCAD REGEN parancsának végrehajtásával (a **command** függvényhívás útján) biztosítható a blokk valamennyi előfordulásának aktualizálása.

Görbe és spline vonalláncok feldolgozása

Ha egy AutoLISP alkalmazásban az **entnext** függvénnyel végiglépked egy Vonallánc csomópontjain, olyan csomópontokkal is találkozhat, amelyeket nem az alkalmazás vagy a felhasználó hozott közvetlenül létre, hanem mint segéd-csomópontokat, a VLEDIT parancs görbe vagy spline opciói illesztettek be automatikusan. Ezeket nyugodtan figyelmen kívül hagyhatja, mivel a rajtuk eszközölt változtatásokat a következő VLEDIT paranccsal végrehajtott görbe- vagy spline-illesztés úgysem veszi tudomásul.

A vonallánc-rajzelem 70-es csoportjának flag-jéből megállapítható, hogy a Vonalláncra illesztett görbe (bitértéke 2) vagy spline-görbe (bitértéke 4). Amennyiben ezen bitek egyike sincs beállítva, a vonallánc valamennyi csomópontja normál, a felhasználó által definiált csomópont. Ha azonban a simulógörbe illesztését jelző bit (2) be van állítva, akkor a vonallánc változó csomópontjainak 70-es csoportjában a bitérték 1 lesz, jelezve, hogy görbeillesztési művelettel kerültek a vonallláncba. Ha az **entmod** függvényt egy ilyen vonallánc

csomópontjainak mozgatására használja azzal a szándékkal, hogy később a VLEDIT parancs segítségével görbét illesszen rájuk, akkor hagyja figyelmen kívül ezeket a csomópontokat.

Hasonlóképpen, ha a vonallánc spline-görbe illesztését jelző bitje (4) be van állítva, akkor találni fog néhány olyan csomópontot, amely flag-jének bitértéke 1 (görbe illesztésével kerültek a csomópontok közé a SPLINESEGS változó negatív értéke mellett), néhány olyat, amelynek bitértéke 8 (spline-illesztésből származnak), az összes többi pedig 16-os bitértékkel szerepel (a spline-görbe keretének vezérlő pontjai). Ez esetben is igaz, hogy, ha az **entmod** függvényt azzal a szándékkal használja a csomópontok mozgatására, hogy később újra spline-görbét illesszen rájuk, akkor elég csak a vezérlő csomópontokat mozgatnia.

Nem-grafikus objektumok kezelése

Az AutoCAD a nemgrafikus-, könyvtár- és szimbólumtábla objektumoknak két típusát használja. Bár vannak hasonlóságok ezen objektumtípusok között, kezelésük eltérő.

Az **entget**, **entmod**, **entdel** és **entmake** függvények minden objektumtípust támogatnak, bár az objektumtípusok önállóan határozzák meg részvételüket ezen függvényekben és részben vagy egészben meghiúsíthatják a végrehajtást. Az AutoCAD beépített objektumaira tekintettel az alábbi szabályok érvényesek (minden, grafikus objektumokra vonatkozó szabály és szigorítás érvényes a nem-grafikus objektumokra is). Nem-grafikus objektumok *nem* adhatóak át az **entupd** függvénynek.

Az **entmake** függvény használata esetén az objektum típusa határozza meg az objektum elhelyezkedését. Például ha az **entmake** egy fólia objektumot kap, akkor az automatikusan a fólia szimbólumtáblába kerül. Ha egy grafikus objektum adódik át az **entmake** függvénynek, akkor az az aktuális térbe (modell- vagy papírtér) kerül.

Szimbólumtábla-objektumok

A szimbólumtáblákra az alábbi szabályok vonatkoznak :

- A szimbólumtábla-bejegyzések inkább néhány megszorítással, mint érvényes rekordábrázolással hozhatóak létre az **entmake** függvény segítségével, így a névellentmondások csak a VPORT táblában fordulhatnak elő. *ACTIVE bemenetek nem hozhatóak létre.

- A szimbólumtábla és a szimbólumtábla-bejegyzés objektum állapota az **entget** függvénnyel kapható meg, a rajzelemnév átadásával. A **tblobjname** függvény egy szimbólumtábla-bejegyzés rajzelemnevének megszerzésére használható.

- Maguk a szimbólumtáblák nem hozhatók létre az **entmake** függvénnyel, bár szimbólumtábla-bejegyzések igen.

- Kezelőcsoportok (5,105) nem változhatnak az **entmod** függvényben, és nem adhatók meg az **entmake** használatával.

- A szimbólumtábla-bejegyzéseknek számos az **entmod** segítségével megváltoztatott mezője lehet. Az **entmod** függvénynek történő átadáshoz egy szimbólumtábla rekordlistának tartalmaznia kell a rajzelem nevét, ami az **entget** függvénnyel - és nem a **tblsearch** vagy **tblnext** segítségével - kapható meg. A szimbólumtábla bejegyzések 70 csoportja figyelmen kívül marad az **entmod** és **entmake** műveletek végrehajtása során.

A VPORT szimbólumtábla kivételével nem megengedett a szimbólumtábla-bejegyzések átnevezése duplázott nevekké. Az alábbi bemenetek nem módosíthatók és nem nevezhetők át. Kivételt képez ez alól, hogy a legtöbb FÓLIA bemenet átnevezhető, a bővített adatok pedig minden szimbólumtábla-bejegyzésen megváltoztathatók.

tábla	*bemenet neve*
VPORT	*ACTIVE
LINESTYLE	CONTINUOUS
LAYER	A bemenetek nem módosíthatók, kivéve a bővített adatokat és az átnevezést

A következő bemenetek nem nevezhetők át, de másképpen módosíthatók, bizonyos megkötéssel:

tábla	*bemenet neve*
LAYER	0
STYLE	STANDARD
DIMSTYLE	STANDARD
BLOCKS	*MODEL_TÉR
BLOCKS	*PAPÍR_TÉR
APPID	A bemenetet nem lehet átnevezni.

Szótárobjektumok

A szótárobjektumokra a következő szabályok érvényesek :

- A szótárobjektumokat az **entget** függvénnyel vizsgálhatja meg, a bővített adatokat pedig az **entmod** függvénnyel módosíthatja. A

szótárbemenetek ugyanakkor nem módosíthatók az **entmod** függvénnyel. A bemenetek elérése csak a **dictsearch** és a **dictnext** függvényekkel lehetséges.

- A szótárbemenetek tartalma nem módosítható az **entmod** függvénnyel, annak ellenére, hogy a bővített adatok módosíthatók.
- Az ACAD* karaktersorozattal kezdődő szótárbemenetek nem nevezhetők át.

Bővített adatok—Xdata

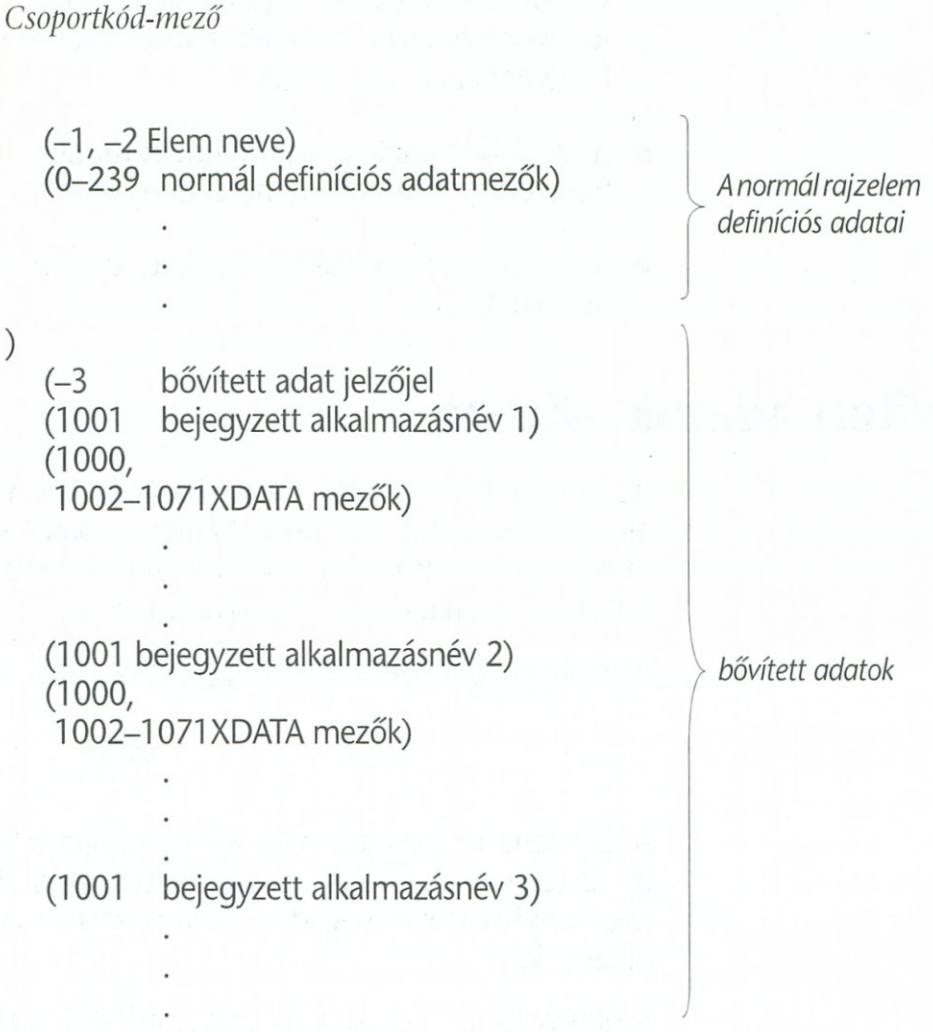
Az AutoLISP számos függvényt kínál az ADS vagy AutoLISP alkalmazások által létrehozott *bővített adatok* kezelésére. Ha egy rajzelem bővített rajzelemadatokkal is rendelkezik, akkor azok a rajzelem normál definíciós adatait követik.

A bővített adatokat kezelő függvények a következők:

regapp xdroom xdsize

A rajzelem bővített adatai az **entget** függvény hívásával kérdezhetők le. Az **entget** függvény egy rajzelem normál definíciós adatait, és a függvényhívásban megadott alkalmazáshoz tartozó bővített adatokat olvassa ki.

Amennyiben az **entget** függvény bővített adatokkal tér vissza, akkor azok kezdetét egy −3-as csoportkód jelöli. A −3-as kód az első 1001-es csoportot megelőző listában szerepel. Az 1001-es csoport tartalmazza az első lekérdezett alkalmazás nevét, mint az az ábrán látható és a következő részben ismertetésre kerül.



Bővített adatok

Bővített adatok szervezése

A bővített adatok egy vagy több 1001-es csoportból állnak, melyek mindegyike egyedi alkalmazásnévvel kezdődik.

Az **entget** függvény által visszaadott bővített adatcsoportok abban a sorrendben követik a definíciós adatokat, ahogyan azok az adatbázisban rögzítésre kerültek. A fenti ábra ezt vázlatosan szemlélteti.

Az egyes alkalmazásokhoz tartozó csoportokon belül az adatok tartalmát, jelentését és szervezését az adott alkalmazás határozza meg. Az AutoCAD csupán megőrzi az információt, de nem használja azt. Amint az ábrán is látható, a bővített rajzelemadatok csoportkódjai

1000-től 1071-ig terjednek. Ezen csoportkódok nagy részéhez ismerős adattípusok tartoznak:

Karakterlánc 1000. A bővített elemadatokban szereplő karakterláncok legfeljebb 255 bájt hosszúak lehetnek (a 256. bájt a null karakter számára van fenntartva).

Alkalmazásnév 1001 (értéke szintén karakterlánc). Az alkalmazásnevek legfeljebb 31 bájt hosszúak lehetnek (a 32. bájt a null karakter számára van fenntartva), és meg kell felelniük a szimbólumtáblákban használt nevekre (például a fólianevekre) érvényes szabályoknak. Az alkalmazásnevek betűket, számokat és $ (dollár), - (kötőjel) vagy _ (aláhúzás) karaktereket tartalmazhatnak, szóközöket viszont nem. Az AutoCAD a név betűit nagybetűkké alakítja.

Fólianév 1003. A bővített rajzelemadathoz tartozó fólia neve.

Adatbázis-sorszám 1005. A rajzelemek sorszáma a rajz adatbázisában.

Három-dimenziós pont 1010. Három valós számérték, amelyek egy pontot alkotnak.

Valós 1040. Egy valós számérték.

Egész 1070. Egy 16-bites egész (előjeles vagy előjel nélküli).

Long 1071. Egy 32 bites előjeles hosszú (`long`) egész szám. Ha az 1071-es csoportban szereplő érték egy rövid (`short`) egész vagy egy valós számérték, akkor azt az AutoCAD hosszú (`long`) egésszé konvertálja; ha az érték érvénytelen (például karakterlánc), akkor a konverzió eredménye egy hosszú nulla (`0L`).

Az AutoLISP az 1071-es csoportokat valós számértékként kezeli. Ha az **`entget`** függvénnyel egy olyan rajzelemet kérdez le, amely 1071-es csoportot tartalmaz, akkor azt valós szám formájában kapja meg. Például:

Megjegyzés Az AutoLISP az 1071-es csoportokat valós számértékként kezeli. Ha az **`entget`** függvénnyel egy olyan rajzelemet kérdez le, amely 1071-es csoportot tartalmaz, akkor azt valós szám formájában kapja meg. Például:

```
(1071 . 12.0)
```

Amennyiben az **`entmake`** vagy az **`entmod`** függvény útján kíván létrehozni egy 1071-es csoportot egy

rajzelemben, az értéket egészként és valósként is megadhatja. Például:

```
(entmake '((..... (1071 . 12) .... )))
(entmake '((..... (1071 . 12.0) .... )))
(entmake '((..... (1071 . 65537.0) .... )))
(entmake '((..... (1071 . 65537) .... )))
```

az AutoLISP azonban változatlanul valós számként adja vissza a csoport értékét:

```
(entmake '((..... (1071 . 65537) .... )))
```

eredménye:

(1071 . 65537.0)

Az ADS az 1071-es csoportokat mindig hosszú (`long`) egészként kezeli.

A bővített rajzelemadatok néhány további csoportja speciális jelentéssel rendelkezik ebben a környezetben (amennyiben az alkalmazás felhasználja őket):

Vezérlő-karakterlánc 1002. Egy bővített adatokat vezérlő karakterlánc "{" vagy "}" lehet: ezek a kapcsos zárójelek lehetővé teszik, hogy az alkalmazások további részlistákba szervezzék adataikat. A bal oldali kapcsos zárójel kezdi a listát, a jobb oldali kapcsos zárójel pedig a legutóbbi listát zárja le; a listák egymásba ágyazhatók.

Megjegyzés Ha egy 1001-es csoport egy listán belül szerepel, akkor azt az AutoCAD egyszerűen karakterláncként kezeli, és nem kezd új alkalmazási csoportot.

Bináris adat 1004. A bináris adatok változó hosszúságú adagokból állnak, melyek az ADS `ads_binary` struktúrájával kezelhetők. Az egyes adagok maximális hossza 127 bájt.

Megjegyzés Az AutoLISP közvetlenül nem képes kezelni a bináris adagokat, így a hosszú (`long`) csoportokra (1071) vonatkozó figyelmeztetés a bináris csoportokra is érvényes.

Világtérbeli pont 1011. Az egyszerű háromdimenziós ponttól eltérően, a VKR-beli pont koordinátái a *szülő* rajzelemmel együtt változnak mozgatás, léptékezés, forgatás és tükrözés során. Ha a NYÚJT parancsot alkalmazza a szülő elemre, ez a pont is elmozdul, amennyiben a kiválasztáshoz használt ablakon belül helyezkedik el.

Világtérbeli elmozdulás 1012. Olyan háromdimenziós pont, amely léptékváltás, forgatás és tükrözés során a szülő elemet

követve változik, NYÚJTás és MOZGATás esetén azonban *nem*.

Világtérbeli irány 1013. Olyan háromdimenziós pont, amely forgatás és tükrözés során a szülő elemet követve változik; léptékváltás, NYÚJTás és MOZGATás esetén azonban *nem*. A VKR-beli irány egy normalizált elmozdulás, amely mindig egységnyi hosszúságú.

Távolság 1041. Olyan valós számérték, amely léptékezés során a szülő elemmel együtt változik.

Aránytényező 1042. Szintén a szülő elemmel együtt léptékezhető valós számérték.

A bővített rajzelemadatok DXF kódjainak ismertetése a 16., "Alkalmazás-specifikus DXF csoportkódok" című fejezetben található meg.

Alkalmazások bejegyzése

Az alkalmazásnevek nemcsak az őket használó rajzelemek bővített adatai között vannak feljegyezve, hanem az APPID táblában is. Minden alkalmazásnak *be kell jegyeznie* az általa használt nevet vagy neveket. Az AutoLISP esetében ez a **`regapp`** függvényhívással történik. A **`regapp`** függvénnyel egy, az alkalmazás neveként használni kívánt karakterlánc adható meg. Amennyiben a nevet sikerült feljegyezni az APPID táblában, a függvény az alkalmazásnévvel, máskülönben `nil` értékkel tér vissza. Ha eredményként `nil` adódik, az általában azt jelzi, hogy a név már szerepel a szimbólumtáblában. Ez valójában nem hiba, hanem egy természetszerűleg adódó visszatérési érték, minthogy az alkalmazás nevét minden rajzban csak egyszer szükséges bejegyezni.

Az alkalmazói programnak az alkalmazásnév bejegyzése előtt célszerű ellenőriznie, hogy a neve nem szerepel-e már az APPID táblában. Ha a név nem található, akkor az alkalmazásnak azt be kell jegyeznie; ellenkező esetben viszont egyszerűen továbbléphet, és használhatja az adatokat, mint azt a következőkben ismertetjük.

A következő részlet a **`regapp`** függvény tipikus felhasználását mutatja be (a **`tblsearch`** függvény ismertetése a "Hozzáférés

szimbólumtáblákhoz és szótárakhoz" című résznél, a 298. oldalon található):

```
(setq alknév "ÉNALK_2356")                    Egyedi alkalmazásnév
(if  (tblsearch "appid" alknév)               Ellenőrzi, be van-e már jegyezve
  (princ (strcat
    "\n" alknév " már be van jegyezve. "))
  (if (=  (regapp alknév) nil)                Egyéb probléma
    (princ (strcat
      "\nAz " appname " alklmazáshoz nem jegyezhetők fel bővített
adatok. "))
  )
)
```

Megjegyzés A **regapp** függvény bizonyos fokú biztonságot nyújt, de nem garantálható, hogy két különböző alkalmazás nem használja ugyanazt a nevet. Ezt oly módon biztosíthatja, ha olyan elnevezési módszert választ, amely a vállalat vagy a termék nevét és egy egyedi számot (például az Ön telefonszámát vagy az aktuális dátumot és időt) használja.

Bővített adatok lekérdezése

Az alkalmazás az általa bejegyzett bővített rajzelemadatokat az **entget** függvény hívásával olvassa ki. Az **entget** függvény definíciós adatokat és a megjelölt alkalmazáshoz tartozó bővített adatokat egyaránt képes szolgáltatni; ehhez az alkalmazás nevét jelölő `alklista` argumentumot is meg kell adni. Az **entget** függvény számára átadott neveknek meg kell felelniük a korábbi **regapp** függvényhívásokkal bejegyzett alkalmazásneveknek. Tartalmazhatnak globális karaktereket is.

A sraffozási minták alapértelmezés szerint bővített adatokat tartalmaznak. A következő programsor ezeknek a bővített adatoknak az asszociációs listáját mutatja be.

Parancs: **(entget (car (entsel)) '("ACAD"))**
Válasszon objektumot: *Válasszon ki egy kapcsolódó sraffozást*

A fenti kódot a parancssorba bebillentyűzve az alábbihoz hasonló listát kap eredményül:

((-1 . <Elem neve: 600000c0>) (0 . "INSERT") (8 . "0") (2 . "*X0")
(10 0.0 0.0 0.0) (41 . 1.0) (42 . 1.0) (50 . 0.0) (43 . 1.0) (70 . 0) (71 . 0)
(44 . 0.0) (45 . 0.0) (210 0.0 0.0 1.0) (-3 ("ACAD" (1000 . "HATCH")
(1002 . "{") (1070 . 16) (1000 . "LINE") (1040 . 1.0) (1040 . 0.0)
(1002 . "}"))))

Az alábbi részlet két megadott alkalmazáshoz tartozó bővített rajzelemadatok lekérdezéséhez szükséges függvényhívások jellegzetes

sorozatát mutatja be. Figyelje meg, hogy az `alklista` argumentumban az alkalmazásnevek lista formájában szerepelnek.

```
(setq munka_elista
  (entget elemnév
    '("ÉN_ALK_1" "VALAMI_MÁS")      csak az "ÉN_ALK-1" és a "VALAMI_MÁS"
                                    alkalmazások bővített adatait olvassa ki
)
(if munka_elista
  (progn
    ...                             a munka-rajzelemlista csoportjának
                                    aktualizálása
    (entmod munka_elista)           csak a munka_elista listában továbbra is
                                    szereplő alkalmazások adatai módosulnak
)
)
```

Ahogy a mintaprogram mutatja, az **entget** függvénnyel kiolvasott bővített rajzelemadatok egy későbbi **entmod** függvényhívással módosíthatók, éppen úgy, ahogyan a normál definíciós adatok (a bővített rajzelemadatok az **entmake** függvénynek átadott elemlistában is definiálhatók).

Az, hogy csak a kért alkalmazásokhoz tartozó bővített adatok szerepelnek a visszaadott listában, védelmet nyújt az ellen, hogy egyik alkalmazás összezavarja a másikhoz tartozó adatokat. Ez kedvező hatással van az alkalmazás memóriaszükségletére, és egyszerűsíti a bővített adatok feldolgozását.

Megjegyzés Mivel az `alklista` argumentumban átadott karakterláncok globális karaktereket is tartalmazhatnak, egy `"*"` alkalmazásnév megadásakor az **entget** függvény a rajzelemhez kapcsolt *összes* bővített adattal fog visszatérni.

Bővített adatok hozzákapcsolása egy rajzelemhez

A bővített rajzelemadatok felhasználásával szinte bármilyen típusú információt tárolhat. Az alábbi egyszerű példa bővített adatok hozzákapcsolását mutatja be egy rajzelemhez.

Rajzoljon egy egyszerű rajzelemet (például vonalat vagy kört), azután billentyűzze be a következőket:

```
(setq lastent (entget (entlast)))  Kiolvassa az utolsó rajzelem definíciós
                                   adatainak asszociációs listáját.
(regapp "ÚJADAT")                  Bejegyzi az alkalmazásnevet.
(setq bővadat                      Elhelyezi az új bővített adatot, ebben az
                                   esetben egy szöveget a bővadat változóban.
 '((-3 ("ÚJADAT"
  (1000 . "Ez valami új!")
  )))
)
(setq újelem
  (append lastent bővadat))        Hozzáfűzi az új adatlistát a rajzelem-listához.
(entmod újelem)                    Módosítja a rajzelemet az új definíciós adatokkal
```

Ha bebillentyűzi a következő utasítást, majd kiválasztja az adott objektumot meggyőződhet róla, hogy az új bővített adatokat valóban hozzákapcsolta a rajzelemhez:

```
(entget (car (entsel)) '("ÚJADAT"))
```

Ez a példa azt az alapvető eljárást mutatja be, amellyel a bővített adatok hozzákapcsolhatók egy rajzelemhez.

A bővített adatok által használt memória kezelése

Jelenleg a bővített rajzelemadatok számára rajzelemenként legfeljebb 16K memória áll rendelkezésre. Mivel egy rajzelem bővített adatait több alkalmazás is létrehozhatja és kezelheti, problémához vezethet, ha a bővített rajzelemadatok megközelítik ezt a határt. Az AutoLISP az **xdsize** és **xdsize** függvényeket kínálja a bővített rajzelemadatok által lefoglalt memória kezeléséhez. Az **xdsize** függvény annak a memóriának a méretét adja vissza (bájtokban), amelyet a számára lista formájában átadott bővített rajzelemadatok lefoglalnának; az **xdsize** függvény pedig az argumentumában megadott rajzelemhez még hozzáfűzhető *szabad* bájtok számával tér vissza.

Az **xdsize** függvénynek egy tetszőleges méretű bővített rajzelem-adatlistát kell végigolvasnia. A függvény lassú lehet, ezért gyakori hívását nem javasoljuk. Célszerűbb ezt a függvényt (az **xdroom** függvénnyel együtt) egy hibakezelő rutinban használni: ha az **entmod** függvényhívás sikertelen, akkor az **xdsize** és **xdroom** függvények segítségével megtudhatja, vajon a hibát az okozta-e, hogy a rajzelemben nem volt elég hely a bővített adatok számára.

Sorszámok a bővített adatokban

A bővített rajzelemadatok sorszámokat (1005-ös csoport) is tartalmazhatnak a rajz belső összefüggéseinek rögzítése céljából. Egyik

rajzelem hivatkozhat egy másikra úgy, hogy annak sorszámát a saját bővített adatai között helyezi el. Ha a sorszámot később kiolvassa a bővített adatokból és a **handent** függvénynek átadja, hozzáférhet a másik rajzelemhez. Mivel egy rajzelemre több másik is hivatkozhat, a bővített adatként tárolt sorszámnak nem szükséges egyedinek lennie. A HLISTA parancs csak azt igényli, hogy a sorszám a bővített adatokban NULL, vagy az aktuális rajzon belül érvényes elemsorszám legyen. A bővített adatként tárolt sorszám érvényességét legegyszerűbben úgy biztosíthatja, ha a hivatkozott rajzelem sorszámát közvetlenül annak definíciós adataiból olvassa ki az **entget** függvény segítségével (ha a sorszámozás be van kapcsolva, akkor a sorszám értéke az 5-ös csoportban található).

Más rajzok rajzelemeire (például azokra, amelyeket az XREF paranccsal illesztett a rajzba) való hivatkozás esetén sorszám (1005-ös csoport) helyett karakterlánc típusú (1000-es csoport) bővített rajzelemadat használatával kerülheti el, hogy a HLISTA parancs hibát jelezzen. A külső referenciaként hivatkozott rajzok elemeinek sorszámai ugyanis vagy *érvénytelenek* az aktuális rajzban, vagy ütköznek az érvényes sorszámokkal. Ha azonban az így beillesztett (XREF Illeszt) rajzelemet véglegesen kell csatolni (XREF Csatol) vagy más módon belefoglalni az aktuális rajzba, akkor ez esetben az alkalmazás dolga a rajzelem-hivatkozások megfelelő módosítása.

Amikor a BEILL, BEILL *, XREF Csatol (XCSATOL) vagy részleges DXFBE paranccsal szerkeszti egybe a rajzokat, akkor a sorszámokat az AutoCAD megváltoztatja úgy, hogy érvényesek legyenek az aktuális rajzban. (Ha a bejövő rajz eredetileg nem tartalmazott sorszámokat, akkor újonnan rendel sorszámokat a beillesztett rajzelemekhez). Ezen parancsok használatakor a beillesztett rajzelemek bővített rajzelemsorszámai *szintén megváltoznak*.

Egy rajzelemnek Blokkdefinícióba történő befoglalásakor (a BLOKK parancs révén) a rajzelem a blokkon belül új sorszámokat kap. (Ha az eredeti rajzelemet a HOPP parancs segítségével visszaállítja, akkor az visszakapja eredeti sorszámait.) Minden sorszám típusú bővített adat értéke változatlan marad. Egy Blokk szétvetésekor (a SZÉTVET parancs révén) a sorszám típusú bővített adatok a rajzok egybeszerkesztéséhez hasonló módon *változnak*. Amennyiben a bővített adatot képező sorszám a blokkon belüli rajzelemre hivatkozik, akkor az új (szétvetett) rajzelem sorszámának értékét kapja meg; ellenkező esetben változatlan marad.

Hozzáférés szimbólumtáblákhoz és szótárakhoz

Az AutoLISP rendelkezik szimbólumtábla- és könyvtárkezelő függvényekkel is. A szimbólumtáblákhoz és a könyvtárakhoz való hozzáférést biztosító függvények a következők:

dictsearch	namedobjdict	tblnext	tblsearch
dictnext	snvalid	tblobjname	

Ebben a részben a **tblnext** és a **tblsearch** függvények alkalmazására talál példákat. A többi szimbólumtábla- és könyvtárelérő függvényekről a 13. fejezetben talál információt. A nem-grafikus objektumokról a 287. oldalon a " Nem-grafikus objektumok kezelése" című részben olvashat.

Szimbólumtáblák

A szimbólumtábla-bejegyzéseket a következő függvényekkel is átalakíthatja.

entdel	entmake	handent
entget	entmod	

A **tblnext** függvény soronként végigpásztázza a szimbólumtábla bejegyzéseit, a **tblsearch** függvény pedig kiolvas egy megadott bejegyzést. A táblanevek megadása karakterlánccal történik. Az érvényes nevek: `"LAYER"`, `"LTYPE"`, `"VIEW"`, `"STYLE"`, `"BLOCK"`, `"UCS"`, `"VPORT"`, `"DIMSTYLE"` és `"APPID"`. Mindkét függvény DXF csoportkódokat tartalmazó listával tér vissza, nagyon hasonlóan az **entget** függvény által szolgáltatott rajzelemadatokhoz.

A **tblnext** függvény első hívásakor a megadott tábla első bejegyzésének adataival tér vissza. Az erre a táblára vonatkozó további hívások a sorban egymás után következő bejegyzések adatait szolgáltatják, hacsak a **tblnext** második argumentuma (*vissza*) nem különbözik `nil`-től, amikor is a **tblnext** függvény újra az első bejegyzés adatait adja vissza.

Az itt látható **BLOKKOLV** függvény az aktuális rajz első blokkjára (ha van ilyen) vonatkozó táblabejegyzést olvassa ki és jeleníti meg lista formájában.

```
(defun C:BLOKKOLV  (/ blk ct)
  (setq blk (tblnext "BLOCK" 1))  kiolvassa az első BLOKK bejegyzést
  (setq ct 0)                     nullára állítja a ct számlálót
  (textpage)                      átvált a szöveges képernyőre
  (princ "\nA BLOKKOLV eredménye: ")
  (repeat (length blk)            annyiszor ismétel, ahány elemű a lista
    (print (nth ct blk))          minden tételt új sorba ír
    (setq ct (1+ ct))             eggyel növeli a számláló értékét
  )
  (princ)                         csendesen lép ki
)
```

A BLOKK táblából kiolvasott bejegyzések egy –2-es csoportot tartalmaznak, amelyben a blokkdefinícióban található első rajzelem neve szerepel. Ha a blokk üres, akkor ez a blokk ENDBLK elemének neve, amely egyébként sosem jelenik meg. Tehát, ha adott egy rajz, és abban egyetlen, DOBOZ nevű blokk, akkor a **BLOKKOLV** hívása a következőket jeleníti meg (az elemnév értéke alkalomról alkalomra változhat):

A **BLOKKOLV** eredménye:
(0 . "BLOCK")
(2 . "BOX")
(70 . 0)
(10 9.0 2.0 0.0)
(–2 . <Elem neve: 40000126>)

Akárcsak a **tblnext** esetében, a **tblsearch** függvény első argumentuma is egy karakterlánc, amely egy táblát nevez meg. A második argumentum azonban egy olyan karakterlánc, amely egy adott szimbólumot nevez meg a táblában. Amennyiben a szimbólum létezik, a **tblsearch** függvény annak adataival tér vissza. Ennek a függvénynek van egy harmadik argumentuma, a *következőt*, amely a **tblnext** függvénnyel végzendő műveletek koordinálására szolgál. Amennyiben a *következőt* értéke `nil`, a **tblsearch** hívásnak nincs hatása a **tblnext** működésére, ellenkező esetben viszont a **tblnext** következő hívása a **tblsearch** által megtalált bejegyzést követő bejegyzésre fog vonatkozni.

A *következőt* opció különösen hasznos a VPORT szimbólumtábla esetében, minthogy egy adott nézetablak-konfigurációban valamennyi nézetablak neve azonos (például *ACTIVE).

Ha a VPORT szimbólumtáblához történő hozzáféréskor a TILEMODE rendszerváltozó ki van kapcsolva, akkor bárminemű változás láthatatlan marad mindaddig, amíg a TILEMODE rendszerváltozót újra be nem kapcsolja. Ne tévessze össze a VPORT szimbólumtáblában

szereplő Vport bejegyzéseket a papírtérbeli Nézetablak (Viewport) rajzelemekkel.

A következő programrészlet a 4NÉZET nevű konfigurációban szereplő minden egyes nézetablakot feldolgozza:

```
(setq v (tblsearch "VPORT" "4NÉZET" T)) megtalálja az első VPORT bejegyzést
while (and v (= (cdr (assoc 2 v)) "4NÉZET"))
  .
  .                                     feldolgozza a bejegyzést ...
  .
  (setq v (tblnext "VPORT"))            veszi a következő VPORT bejegyzést
)
```

Szótárak

A szótár olyan tárolóobjektum amely funkciójában a szimbólumtáblákhoz hasonlít. A **dictsearch** és a **dictnext** szótárbejegyzéseket kezelő függvények. Minden egyes szótárbejegyzés tartalmaz egy szövegnév kulcsszót, valamint egy kötött, felhasználói, a bemenetre vonatkozó sorszám referenciát. A szótárbejegyzéseket a **entdel** függvényhez közvetlenül kapcsolódó objektumbemenetek neveivel távolíthatja el. A szövegnév kulcsszóra ugyanaz a szintaxis és karakterhasználat érvényes, mint a szimbólumtábla-nevekre.

10

PÁRBESZÉDPANELEK KEZELÉSE

Az AutoCAD segítségével a felhasználók párbeszédpaneleket tervezhetnek az alkalmazásaikkal való használatra. Egy párbeszédpanel megjelenését a párbeszédvezérlő-nyelv (DCL) fájlok határozzák meg, amelyek leírása ezen kézikönyv III., „Programozható párbeszédpanel referencia" című részében található. A párbeszédpanel működését egy AutoLISP vagy ADS alkalmazás irányítja. Ez a fejezet bemutatja a párbeszédpanelek vezérlését az AutoLISP segítségével. Bár a fejezetben található néhány példa a DCL fájlokra, elolvasása előtt segítségre lehet, ha tanulmányozza a III. részt.

Párbeszédpanelek megnyitása és bezárása

Az AutoLISP az alábbi függvényekkel rendelkezik párbeszédpanelek megnyitására és bezárására.

done_dialog	new_dialog	term_dialog
load_dialog	start_dialog	unload_dialog

Mielőtt a párbeszédpanel függvényeket használ, léteznie kell egy DCL fájlnak, ami definiálja a párbeszédpanelt. Mentse el az alábbi DCL programot egy *hello.dcl* nevű fájlba, egy a support elérési útvonalon lévő alkönyvtárba. Ez a DCL fájl egy Minta-párbeszédpanel címkéjű párbeszédet definiál, ami egy szöveges zónát és egy OK nyomógombot tartalmaz.

```
sample: dialog {
  label = "Minta-párbeszédpanel";
  : text { label = "Helló világ!"; }
  ok_only;
}
```

Egy párbeszédpanel megjelenítése számos lépésből áll. Az első lépésben a **load_dialog** függvénnyel a DCL fájlt a memóriába kell tölteni; így megkapja a DCL azonosító számot. A DCL fájl betöltése után meg kell hívni a **new_dialog** függvényt, és paraméterként át kell adni neki a párbeszédnevet illetve a DCL azonosító számot. Ha a **new_dialog** nem nil értékkel tér vissza, hívja meg a **start_dialog** függvényt, hogy a vezérlés visszakerüljön az AutoCAD programhoz és a felhasználóhoz. Mivel az `ok_only` zóna (az OK nyomógomb) előredefiniált zóna, viselkedése is előre meghatározott. A felhasználó általában még a **start_dialog** függvény hívása előtt hozzárendeli a műveleteket, és beállítja a zónák állapotát és értékét (további tudnivaló a "Zónakifejezések és visszajelzések" című résznél, a 305. oldalon). Az `ok_only` zónához rendelt művelet a **done_dialog**. Amikor kiválasztja az OK nyomógombot, az AutoCAD a **done_dialog** hívást átadja az AutoLISP alkalmazásnak és befejezi a párbeszédet. Ezután Ön átadja a DCL azonosító számot az **unload_dialog** fügvénynek, ami kitörli a DCL fájlt a memóriából. (A párbeszédpanel függvénysorozatának teljes leírását megtalálja a "Függvényhívások sorrendje" című résznél, a 324. oldalon)

Ha elmentette a *hello.dcl* fájlt egy a support elérési úton lévő alkönyvtárba, az alábbi AutoLISP függvényt használhatja a fájl

megjelenítésére. (Ebben a változatban a hibaellenőrzés minimális, hogy a felépítés jobban látható legyen.)

```
(defun c:hello ( / dcl_id )
  (setq dcl_id (load_dialog "hello.dcl"))   Betölti a DCL fájlt
  (if (not (new_dialog "hello" dcl_id))     Inicializálja a párbeszédet
    (exit)                                  Kilép, ha nem működik
  )
  (start_dialog)                            Megjeleníti a párbeszédpanelt
  (unload_dialog dcl_id)                    Kitörli a DCL fájlt
  (princ)
)
```

A **start_dialog** hívás aktív marad mindaddig, amíg a felhasználó ki nem választ egy olyan zónát (általában egy nyomógombot), amelynek hozzárendelt akciókifejezése meghívja a **done_dialog** függvényt. A **done_dialog** hívást a zóna explicit módon is kiadhatja; és kiadható a kiválasztott zóna `is_cancel` attribútumának `true` logikai értékre állításával is.

Figyelmeztetés Elméletileg a **start_dialog** függvény hívásakor a párbeszédpanel átveszi a bevitel irányítását, de a Windows és más platformok esetében ez csak a **new_dialog** függvény esetében történik meg. Ezt nem kell figyelembe venni a programok írásánál. Ha viszont interaktív módon használja ezeket a függvényeket, azokat egyetlen utasításként kell beírni az AutoCAD Parancs promptjához. Írja be őket egy **progn** vagy más függvénybe. Ha ezt nem teszi meg, a **new_dialog** interaktív hívása lefagyaszthatja a képernyőt. A **new_dialog** és a **start_dialog** függvények interaktív hívása hibakeresés alatt lehet hasznos. (Ezen függvények interaktív használatára lásd a "DCL hibakezelés" című résznél, a 624. oldalon.)

Ameddig egy párbeszédpanel aktív -- vagyis a **start_dialog** hívás ideje alatt -- az alább felsorolt AutoLISP függvények *nem* hívhatóak meg. Ezek a függvények vagy hatással vannak a megjelenítése, ami nem változhat meg, ameddig a párbeszédpanel látható, vagy olyan felhasználói adatbevitelt igényelnek, ami nem foglalja magába a párbeszédpanelt.

command	getdist	grclear	prompt[a]
entdel	getint	grdraw	redraw
entmake	getkword	grread	ssget[b]
entmod	getorient	grtext	textpage
entsel	getpoint	grvecs	textscr
entupd	getreal	menucmd	
getangle	getstring	nentsel	

getcorner	graphscr	osnap

a. *A szöveget kiíró függvények, mint a* **print, princ** *és* **prin1** *használhatók hibakereső információk kiírására párbeszédpanelek tesztelése során. Ne használja ezeket végleges termékekben. Ha a párbeszédpanel lefedi a Parancs prompt területet, a fenti függvények át fogják írni a párbeszédpanel ezen részét.*
b. *Az* **ssget** *opciók interaktív használata nem megengedett, de más opcióké igen.*

Ha a program meghív egyet a fenti függvények közül a **`start_dialog`** és a **`done_dialog`** függvényhívások között, az AutoCAD megszakít minden párbeszédpanelt és kiírja az alábbi hibaüzenetet:

Az AutoCAD visszautasította a függvényt

Ha a CMDACTIVE rendszerváltozó értéke 7-nél nagyobb, akkor a párbeszédpanel aktív. A CMDACTIVE rendszerváltozónak bit-kódolású értékei vannak, amelyek jelzik a parancs-, forgatókönyv- és párbeszédpanel-aktivitást. További tudnivalók az *AutoCAD on-line súgójában.*

Amint a táblázat második lábjegyzetéből kiderül, az **`ssget`** hívások megengedettek, kivéve az interaktív opciókat.

Ha felhasználónak olyan adatot kell megadnia, amely egy grafikus ablakot használ fel, és nem magát a párbeszédpanelt (például egy pont megadása vagy egy objektum kijelölése), akkor a párbeszédpanelt *el kell rejteni.* Vagyis meg kell hívnia majd a **`done_dialog`** függvényt, hogy a grafikus ablak újra látható legyen, később pedig -- miután a felhasználó elvégezte a kiválasztást -- újraindítani a párbeszédpanelt. További információt talál "Párbeszédpanelek elrejtése" című résznél, a 314. oldalon.

A **`term_dialog`** függvény megszakít minden aktuális párbeszédpanelt úgy, mintha a felhasználó zárta volna le őket. Ez a függvény használható az egymásba ágyazott párbeszédpanelek sorozatának lezárására.

Zóna- és attributumkezelés

Az AutoLISP a következő függvényeket kínálja az zónák és attributumok kezelésére.

action_tile	get_tile	set_tile
get_attr	mode_tile	

Az ide vonatkozó példákat a jelen fejezetben, illetve a 13., "AutoLISP függvények katalógusa" című fejezetben találja meg.

Zónakifejezések és visszajelzések

Annak meghatározásához, hogy milyen akció hajtódjon végre, amikor egy párbeszédpanel egy bizonyos zónáját kiválasztja, egy AutoLISP kifejezést kell rendelni az adott zónához az **action_tile** függvény segítségével. Ezt *akciókifejezésnek* hívjuk. Az akciókifejezésen belül gyakran szükséges a hozzáférés a DCL-fájlbeli attribútumokhoz. Ezt a **get_tile** és a **get_attr** függvények teszik lehetővé. A **get_attr** függvény visszatérési értékei a DCL fájlban lévő, felhasználó által definiált attribútumok. A **get_tile** függvény megadja egy zóna pillanatnyi, *futásidő alatti* értékét az adott zónára vonatkozó felhasználói bevitel alapján. Az akciókifejezéseket a **new_dialog** függvényhívás után, és a **start_dialog** hívás előtt kell definiálni.

A felhasználó által kiválasztott zónáról, illetve a zóna tartalmának módosításáról szóló információ *visszajelzésként* tér vissza az akciókifejezéséhez. A legtöbb esetben a párbeszédpanel minden aktív zónája egy visszajelzést fog kiváltani. Az akciókifejezés visszajelzésre adott válaszának (*visszajelzési függvény*) ellenőriznie kell a hozzá tartozó zóna érvényességét, és aktualizálnia kell minden a zóna értékére vonatkozó információt a párbeszédpanelben. A párbeszédpanel aktualizálásához tartozhat egy hibaüzenet kiadása, más zónák letiltása, és a megfelelő szöveg megjelenítése egy szerkesztőablakban vagy listaablakban.

Csak az OK nyomógomb (vagy annak megfelelője) kérdezze le ténylegesen a zónák értékét abból a célból, hogy elmentse a felhasználó által utoljára beállított értékeket. Más szóval, a zóna értékéhez kapcsolódó változókat az OK nyomógombhoz tartozó visszajelzésen belül aktualizálja, és ne egyenként, a zónákhoz tartozó visszajelzéseken belül. Amennyiben az egyes zónákhoz tartozó visszajelzéseken belül állandó változók kerülnek aktualizálásra, akkor nincs mód az értékek visszaállítására abban az esetben, ha a felhasználó az Mégsem nyomógombot nyomja meg. Ha az OK nyomógomb visszajelzése hibát észlel, akkor jelenítsen meg egy hibaüzenetet, a fókuszt irányítsa a hibás zónára, és ne lépjen ki a párbeszédpanelből.

Amennyiben a párbeszédpanel több olyan zónát tartalmaz, amelyek kezelése hasonló, kényelmesebb lehet ezekhez a zónákhoz egyetlen visszajelzési függvényt társítani. Ebben az esetben is érvényes az az elv, hogy ne véglegesítse a változásokat mindaddig, amíg a felhasználó az OK nyomógombbal ezt nem kéri.

Az **action_tile** hívásán kívül még két módja van az akciók definiálásának. Amikor meghívja a **new_dialog** függvényt, definiálhat egy *alapértelmezés* szerinti akciót az egész párbeszédpanelre vonatkozóan, továbbá a zóna `action` attribútuma útján szintén definiálhat egy akciót. Ezek alternatív eszközök az akciók definiálására; előfordulásuk sorrendjét az "Alapértelmezés szerinti és DCL akciók" című résznél, a 309. oldalon ismertetjük.

Akciókifejezések

Az akciókifejezések az alábbi táblázatban látható változókhoz férnek hozzá. Ezek jelzik a kiválasztott zónát és annak állapotát az akció időpontjában. Ezek a változónevek foglaltak, értékük csak olvasható; csak az akciókifejezésből kiolvasva van jelentésük.

Akciókifejezések változói

Változó	*Leírás*
`$key`	A kiválasztott zóna key attribútuma. A változó minden akcióra vonatkozik.
`$value`	A zóna aktuális értéke karakterlánc formájában, mint például egy szerkesztőablak vagy egy kapcsoló "1" vagy "0" karakterlánca. A változó minden akcióra vonatkozik. Ha a zóna egy listaablak (vagy legördülő lista) és nincs kiválasztott tétel, a `$value` változó nil értékű lesz.
`$data`	Az alkalmazás által kezelt adat (ha létezik), amely közvetlenül a **new_dialog** hívása után került beállításra a **`client_data_tile`** útján. Ez a változó minden akcióra vonatkozik, de a `$data`-nak nincs jelentése, kivéve ha az alkalmazás a **`client_data_tile`** meghívásával már inicializálta azt. További tudnivaló az "Alkalmazás-specifikus adatok" című résznél, a 322. oldalon.
`$reason`	Az indítékkód, amely megmutatja, hogy a felhasználó mivel váltotta ki az akciót. Az edit_box, list_box, image_button és slider zónák esetében használatos Ez a változó jelzi, miért következett be az akció. Értéke bármely akcióra be lehet állítva, de csak akkor kell megvizsgálnia, amikor az akció hozzá van kapcsolva az `edit_box`, `list_box`, `image_button`, vagy `slider` zónákhoz. További tudnivalók a "Visszajelzési indítékok" című résznél, a 307. oldalon találhatók.
`$x` `$y`	Egy image_button kiválasztásának *x vagy y* koordinátája Ezek olyan karakterláncok, melyek az `image_button` zónából a felhasználó által kiválasztott pont X és Y koordinátáit tartalmazzák. Más zónáknál nincs jelentésük. Az *X* koordináta a zónára vonatkozóan a *`(dimx_tile)`*, az *Y* pedig a *`(dimy_tile)`* által visszaadott tartományba esik. Ezeknek a koordinátáknak a felhasználására vonatkozóan az "Ábra-nyomógombok kezelése" című résznél, a 322. oldalon talál példát.

Amennyiben a szerk1 egy szerkesztőablak, az alábbi (**action_tile**) hívásban megadott akciókifejezés kerül kiértékelésre, amikor a felhasználó elhagyja a szerkesztőablakot:

(**action_tile** "edit1" "(setq ns $value)")

Ebben az időpontban a $value a felhasználó által a szerkesztőablakba beírt karakterláncot tartalmazza, amit a kifejezés az ns változóba ment el.

A következő példa a kiválasztott zóna nevét menti el azért, hogy a program hivatkozhasson rá:

(**action_tile**"edit1" "(setq newtile $key)")

A newtile változó a kiválasztott zóna nevét, ebben az esetben a "edit1" karakterláncot fogja tartalmazni. A $key változónak leginkább egy olyan függvényben láthatja hasznát, amely több különböző zóna akciójaként szolgál.

Ha a zónát több **action_tile** is meghívta, csak az utolsó (a **start_dialog** függvényt megelőző) az érvényes. (Hasonlóan ahhoz, mintha több értéket adna ugyanannak a változónak.) A PDB (programozható párbeszédpanel) csak egy műveletet tesz lehetővé zónánként.

Visszajelzési indítékok

A visszajelzési *indíték*, amelyet az AutoLISP esetében a `$reason` változó, az ADS esetében pedig a visszajelzési csomag indítékmezője szolgáltat, az akciót, illetve a visszajelzést kiváltó okot adja meg. Ez az érték az akciók minden fajtája esetén beállításra kerül, Önnek azonban csak akkor kell megvizsgálnia, ha az akció egy `edit_box`, `list_box`, `image_button`, vagy `slider` zónához tartozik. A lehetséges értékeket a következő táblázat mutatja. A táblázatot követő szöveges rész a 2-es, 3-as és 4-es kódot írja le. Az 1-es kód részletes leírása a táblázatban szerepel.

Visszajelzési indítékkódok

Kód	*Leírás*
1	Ez a legtöbb fajta akciózóna esetében előforduló érték. Azt jelenti, hogy a felhasználó kiválasztotta a zónát; például rámutatással (vagy az alapértelmezés szerinti zóna esetében a ↵ lenyomásával, amennyiben a platform felismeri a gyorsító billentyűket).
2	Szerkesztőablakok: A kód jelzi, hogy a felhasználó kilépett a szerkesztőablakból, de nem végzett végső kiválasztást.

Visszajelzési indítékkódok

Kód	Leírás
3	Csúszkák: A kód jelzi, hogy a felhasználó megváltoztatta a csúszka értékét a mutató vontatásával , de nem végzett végső kiválasztást.
4	Listaablakok és ábra-nyomógombok: Ez a visszajelzési indíték mindig az 1-es kód után következik. Általában azt jelenti: "erősítse meg az előző kiválasztást." Ennek soha nem szabad érvényteleníteníe az előző kiválasztást, mert az zavarja és bosszantja a felhasználót.

Kód 2—Szerkesztőablakok

A felhasználó kilépett a szerkesztőablakból — például a Tab billentyűvel, illetve egy másik zónára kattintva —, de nem végzett végső kiválasztást. Ha egy szerkesztőablak esetében ez a visszajelzés indítéka, akkor az alkalmazásnak nem szabad aktualizálnia a hozzá tartozó változó értékét, viszont jó alkalom a szerkesztőablak értékének ellenőrzésére.

Kód 3—Csúszkák

A felhasználó megváltoztatta a csúszka értékét a mutató vontatásával (vagy ezzel egyenértékű művelettel), de nem végzett végső kiválasztást. Ha ez a visszajelzés indítéka egy csúszka esetében, akkor az alkalmazásnak nem szabad aktualizálnia a hozzá tartozó változó értékét; elképzelhető viszont, hogy a csúszka helyzetét jelző szöveget aktualizálnia kell (mint azt a "Csúszkák" című résznél, a 645. oldalon ismertettük). Idevonatkozó programozási példát a "Csúszkák kezelése" című résznél, a 312. oldalon talál.

Kód4—Listaablakok

A felhasználó duplán rákattintott a listaablakra. A dupla kattintás jelentése az alkalmazástól függ. Ha a párbeszédpanel fő célja egy tétel kiválasztása a listában, akkor a dupla kattintás jelenthet egy kiválasztást, majd a párbeszédpanel elhagyását (ebben az esetben a list_box zóna is_default attribútuma igaz kell hogy legyen). Ha a listaablak nem az elsődleges zóna a párbeszédpanelben, a dupla kattintást egy kiválasztás elvégzésével egyenértékűen kell kezelni (1-es kód).

Azok a listaablakok, amelyek lehetővé teszik, hogy a felhasználó több tételt is kiválasszon, *nem* támogathatják a dupla kattintást.

Kód4—Ábra-nyomógombok

A felhasználó duplán rákattintott az ábra-nyomógombra. A dupla kattintás jelentése az alkalmazástól függ. Gyakran egyetlen kattintás elég a nyomógomb kiválasztásához; sokszor jobb azonban, ha egy

kattintás (vagy billentyű leütése) csak kiemelt megvilágításba helyezi a nyomógombot, majd egy ↵ vagy dupla kattintás választja ki azt.

Alapértelmezés szerinti és DCL akciók

Egy akció meghatározásának nem az **action_tile** függvény az egyetlen eszköze. Egy zóna DCL leírása tartalmazhat egy AutoLISP kifejezést magában foglaló `action` attribútumot, továbbá a **new_dialog** hívás meghatározhat egy alapértelmezés szerinti akciót a párbeszédpanel egészére. Egy zónához azonban egyszerre csak egy akció tartozhat. Amennyiben a DCL és az alkalmazás egynél több akciót határoz meg, ezek a következő prioritási sorrend szerint hatálytalanítják egymást:

1 A **new_dialog** hívás által meghatározott alapértelmezés szerinti akció (csak akkor használatos, ha a zónához nincs külön akció hozzárendelve)

1 A DCL fájlban az `action` attribútum által meghatározott akció (*csak az AutoLISP esetében*)

1 Az **action_tile** hívás által hozzárendelt akció (ennek a legmagasabb a prioritása).

Zónák kezelése

A program inicializáláskor és akció (visszajelzés) idején egyaránt befolyást gyakorol az aktuális párbeszédpanel zónáira. A következő rész a zónákat kezelő függvények használatába vezet be és bemutatja, hogyan kell inicializálni és módosítani a zónamódokat és -értékeket.

Zónamódok és zónaértékek inicializálása

A zóna inicializálása a következőket foglalja magában:

- A párbeszédpanel kezdeti billentyűzet beállítása
- Engedélyezés vagy letiltás
- Szerkesztőablak vagy ábra esetén a zóna tartalmának kijelölése

Ezen műveletek végrehajtása a **mode_tile** meghívásával történik A zóna értékét a **set_tile** használatával állíthatja be.

Egy alapértelmezés szerinti értéknek — például egy vezetéknévnek — egy szerkesztőablakban történő megjelenítéséhez, és a párbeszédpanel kezdeti fókuszának ráirányításához a következő programrészletet használhatja:

```
(setq name _string "Kovács")        Alapértelmezés

( set_tile "veznév" name_string)    Inicializálja a mezőt
( mode_tile "veznév" 2)             A 2 a zónára irányítja a fókuszt
```

Egy további **mode_tile** hívás kiemelt módba helyezi a szerkesztőablak egész tartalmát, így a felhasználó azonnal felülírhatja az alapértelmezés szerinti tartalmat. Például:

```
(mode_tile "veznév" 3)  A 3 kiválasztja a szerkesztőablak tartalmát
```

Megjegyzés Egyes platformok esetében a fókusznak egy szerkesztőablakra történő ráirányítása automatikusan kiemeli azt, s így ez utóbbi lépés feleslegessé válik (bajt azonban nem okoz).

Állapotok és értékek megváltoztatása a visszajelzés során

A visszajelzés során ellenőrizheti a zóna értékét. Amennyiben az alkalmazás megkívánja, a **set_tile** újbóli hívásával módosíthat az értéken. Visszajelzés közben a **mode_tile** segítségével a zóna állapotát is megváltoztathatja. Az alábbi táblázatban a **mode_tile** *mode* argumentumának lehetséges értékei láthatók:

Zónaállapot-kódok a mode_tile függvényekhez

Érték	*Leírás*
0	Zóna engedélyezése
1	Zóna letiltása
2	A fókusz ráirányítása a zónára
3	Szerkesztőablak tartalmának kiválasztása
4	Ábra kiemelésnek be- vagy kikapcsolása

Megjegyzés Amennyiben a **mode_tile** segítségével letiltja azt a zónát, amelyre a fókusz aktuálisan irányul, akkor egy újabb **mode_tile** hívással a fókuszt egy másik zónára (legtöbbször a párbeszédpanel következő tabulátorhelyére) kell átirányítania. A fókusz máskülönben a letiltott zónán maradna, ami nemcsak ésszerűtlen, de hibát is okozhat.

Jó példa a zónák "önmaguk általi letiltására", amikor a "többoldalas" párbeszédpanelt a felhasználó a Következő, illetve az Előző nyomógombok kijelölésével végiglapozza. A Következő nyomógomb letiltásra kerül, amikor a felhasználó az utolsó előtti oldalon nyomja azt meg. Ugyanez történik, amikor az Előző nyomógombot a második oldalon nyomja meg. A programnak mindkét esetben le kell tiltania azt a nyomógombot, amelyet megnyomtak, majd a fókuszt egy másik zónára kell átirányítania.

Tételezzük fel, hogy a "group_on" nevű zóna egy kapcsoló, amely egy "group" nevű csoportot vezérel; amikor a kapcsoló kikapcsolt állapotban van, akkor a csoport legyen letiltva, s ne legyen módosítható. Ebben az esetben a kapcsolóhoz a következő akciót rendelheti hozzá. (Figyelje meg, hogy a \" vezérlőkarakter, megengedi az időzőjelek használatát az **action_tile** függvényben.)

```
(action_tile "group_on" "(mode_tile \"group\" (- 1 (atoi
$value)))")
```

Az akciókifejezésben szereplő kivonás és **(atoi)** hívás a **(mode_tile)** függvény *mode* argumentumát állítja be. Mivel a kapcsoló értéke kikapcsolt állapotban nulla, bekapcsolt állapotban pedig 1, a kivonás megfordítja az értékét, a mode argumentum pedig azt szabályozza, hogy a csoport engedélyezett-e vagy sem.

A `value` mellett a zóna más attribútumait is hasznos megvizsgálni. Ezt a **get_attr** függvény segítségével teheti meg. Tegyük fel például, hogy egy "nyomjmeg" nevű nyomógomb címkéjét kívánja megtudni:

```
(get_attr "nyomjmeg" "címke")
```

Megjegyzés Amennyiben a **get_attr** útján egy value attribútumot kérdez le, akkor a DCL fájlban megadott `value` attribútumot (a zóna kezdeti értékét) fogja megkapni. A **get_tile** függvény pedig megadja a zóna futásidőbeli értékét. A két érték nem feltétlenül egyezik meg.

A **get_attr** függvény a kért attribútum értékével mint karakterlánccal tér vissza.

Rádiócsoportok kezelése

A rádiógombok rádiócsoportokon belül találhatók. Az egyes rádiógombok értéke "1" (Be) vagy "0" (Ki) lehet, a rádiócsoport értéke pedig az aktuálisan kiválasztott nyomógomb key attribútuma. A csoporton belül a rádiógombok értékét a programozható párbeszédpanelek (PDB) csomag kezeli, amely azt is biztosítja, hogy egyszerre csak egy legyen bekapcsolva közülük. Az alkalmazásban minden egyes rádiógombhoz külön is hozzárendelhet egy akciót, de kedvezőbb lehet a rádiócsoport egészéhez társítani egy akciót, majd a csoport value attribútumából megállapítani, hogy éppen melyik rádiógomb került kiválasztásra.

Tételezzük fel, hogy egy rádiócsoport azt szabályozza: egy háromdimenziós tárgy melyik nézete jelenjen meg — miután a felhasználó elhagyja a párbeszédpanelt. A csoport négy rádiógombot tartalmaz (lehet természetesen több is):

```
(action_tile "view_sel" "(pick_view $value)")
   .
   .
   .
(defun pick_view (which)
```

```
    (cond
      ((= melyik "elöl") (setq show_which 0))
      ((= melyik "felül") (setq show_which 1))
      ((= melyik "bal") (setq show_which 2))
      ((= melyik "jobb") (setq show_which 3))
    )
)
```

Ebben a példában az összes rádiógombhoz egyetlen változó tartozik, amely különböző értékeket vehet fel. Ezek a változók akciókat is előidézhetnek, így például a párbeszédpanel más választási lehetőségeinek a letiltását. Egy nagyméretű rádiócsoport esetében kedvezőbb lehet egy táblázatban tárolni a hozzá tartozó értékeket. Amennyiben táblázatot használ, akkor azt úgy szerkessze meg, hogy független legyen a nyomógomboknak a csoporton belüli sorrendjétől. A PDB csomag ugyanis nem köti ki, hogy a DCL fájl módosulása esetén nem változhat a sorrend.

Csúszkák kezelése

A csúszkákhoz tartozó akciók, illetve visszajelzések kezelésekor az alkalmazásnak meg kell vizsgálnia a visszajelzéstől kapott *indítékot*. Az indítékkódot nem feltétlenül szükséges megvizsgálni, de célszerű, mert ezzel csökkenthető az alkalmazás által elvégzendő feldolgozás. A csúszka által generált visszajelzések gyakorisága a platformtól függ; néhány platform esetében a csúszka által észlelt *minden* egérmozdulat kivált egy 1-es visszajelzést.

Az alábbi példa egy csúszka kezelését végző függvény alaptípusát mutatja be. A függvényt egy csúszkához rendelt akciókifejezésből történő hívásra szántuk. A függvény által használt `slider_info` zónának az a rendeltetése, hogy a csúszka aktuális értékét decimális számformában jelenítse meg. Erre a célra gyakran szerkesztőablakot szoktak használni, ami választási lehetőséget kínál a felhasználónak, hogy a csúszkát mozgatja, vagy közvetlenül beírja az értékét. Amennyiben a felhasználó az értéket a `slider_info` szerkesztőablakba billentyűzi be, akkor a szerkesztőablak visszajelzésének aktualizálnia kell a csúszka értékét (és viszont).

```
(action_tile
  "csúszka"
  "(slider_action $value $reason)"
)
(action_tile
  "slider_info"
  "(ebox_action $value $reason)"
)
  .
  .
  .
(defun slider_action(val why)
```

```
  (if (or (= why 2) (= why 1))
  (set_tile "slider_info" val)     Mutatja a közbenső eredményt
  )
)

(defun ebox_action(val why)
  (if (or (= why 2) (= why 1))
  ( set_tile "csúszka" val)        Mutatja a közbenső eredményt
  )
)
```

Szerkesztőablakok kezelése

A szerkesztőablakok kezelésére szolgáló akciók és visszajelzések nagyon hasonlóak a csúszkákéhoz, azzal a különbséggel, hogy mivel a szerkesztőablakban a karakterek eleve láthatók, így nincs szükség közbenső eredményekkel kapcsolatos tevékenységre. A szerkesztőablakok csak akkor térnek vissza visszajelzéssel, ha az adott zónára vonatkozó fókusz elveszett.

A program ellenőrzi az értéket, de nem jeleníti meg újra:

```
(action_tile "szerkabl1" "(edit_action $value $reason)")
  .
  .
  .
(defun edit_action (val miért)
  (if (or (=why 2) (= why 4))
    .                          Az átmeneti érték ellenőrzésének helye
    .
  )
)
```

Párbeszédpanelek egymásba ágyazása

Beágyazott párbeszédpaneleket úgy készíthet és kezelhet, hogy egy akció-kifejezésen vagy visszajelzési függvényen belül egyszerűen meghívja a **new_dialog** és a **start_dialog** függvényt. Ahhoz például, hogy a függvény megjelenítse a "Helló világ" párbeszédpanelt akkor, amikor a felhasználó kiválasztja az `button_1` elnevezésű nyomógombot, a következő utasítást kell elhelyezni az alkalmazásban:

```
(action_tile "button_1" "(c:hello)")
```

A felhasználónak ki kell lépnie a beágyazott párbeszédpanelből ahhoz, hogy újra az előzőt használhassa.

Megjegyzés A túl sok beágyazott párbeszédpanel kényelmetlen, sőt bosszantó lehet a felhasználó számára. Az AutoCAD nyolcban szabja meg a határ; azt javasoljuk azonban, hogy négyszeres mélységnél jobban ne ágyazza egymásba a párbeszédpaneleket.

Párbeszédpanelek elrejtése

Amíg egy párbeszédpanel aktív, a felhasználó nem végezhet interaktív kiválasztást; ezért ha azt kívánja, hogy a felhasználó a grafikus képernyőn valamit kiválasszon, ahhoz előbb el kell *rejtenie* a párbeszédpanelt, azután pedig vissza kell állítania. A párbeszédpanelt elrejteni ugyanaz, mint a **done_dialog** segítségével lezárni, azzal a különbséggel, hogy a visszajelzési függvényben meg kell adni a **done_dialog** *status* argumentumát annak jelzésére, hogy a párbeszédpanel elrejtése — nem pedig befejezése vagy elvetése — következett be. A *status* argumentumot egy, az alkalmazásban definiált értékre állítsa be. A **start_dialog** függvény a párbeszédpanel eltűnésekor ezzel a *status* értékkel tér vissza. A programnak ezután meg kell vizsgálnia a **start_dialog** által visszaadott `status` értéket, hogy az ezután szükséges lépést megállapítsa. Ez azt is jelenti, hogy Önnek a hívást ciklusba foglalva "csapdába kell ejtenie" a **start_dialog** visszatérési értékét, hogy később ismét megjeleníthesse az elrejtett párbeszédpanelt. (A szabvány és az alkalmazás szerinti `status` értékekre vonatkozóan lásd a "**done_dialog**" címszót a 13. fejezetben.)

A *bmake.lsp* mintaprogramban szereplő Pontkijelölés nyomógomb elrejti a párbeszédpanelt abból a célból, hogy a felhasználó kijelölhessen egy pontot a grafikus képernyőn. A nyomógomb kijelölésekor bekövetkező akció hatására a párbeszédpanel lezárul, és a `status` a speciális 4-es értéket veszi fel:

```
(action_tile "pick_pt" "(done_dialog 4)")
```

Ezután a **start_dialog** függvény visszatér, a program megvizsgálja a visszaadott státuszértéket, és ha szükséges, aktualizál egy változót:

```
(setq what_next (start_dialog))
(cond                          Döntse el, mi a következő tennivaló
                               Ha a bázispont kijelölését kérte...
  ((= what_next 4)
    (setq pick_pt (getpoint "Beillesztés bázispontja: "))
    (setq x_pt (rtos (car pick_pt) 2 4))
    (setq y_pt (rtos (cadr pick_pt) 2 4))
    (setq z_pt (rtos (caddr pick_pt) 2 4))
  )
)
```

A fő függvény lényegi része egy ciklusban helyezkedik el, így a **new_dialog** és a **start_dialog** függvények (a **action_tile**, **set_tile**, **start_list** és hasonló függvényekkel együtt) hívása addig ismétlődik, amíg a felhasználó meg nem nyomja az OK vagy

Mégsem nyomógombot. A következő példa része a *bmake.lsp* példaalkalmazásnak.

```
(defun c:bmake ...
                                    Töltse be a párbeszédpanelt és végezze el az inicializást
  (while (< 2 mi_követk)            Az állapot jelzi a rejtést
                                    Egyéni párbeszédpanelek inicializálása, mint pl.
                                    new_dialog, action_tile, set_tile,
                                    és start_list meghívása
    (setq what_next
      ( start_dialog)
    )
    (cond
                                    Vizsgálja meg, hogy a MI_KÖVETK, azt jelzi-e, hogy a
                                    párbeszédpanel rejtett, majd hajtsa végre a szükséges
                                    műveletet
    )
    .
    .
    .
  )
)
```

Beágyazott párbeszédpanel esetében az eljárás bonyolultabb, de alapvetően azonos:

```
(defun c:maindlg ( / what_next what_next1 )
  (setq what_next 5)
  (if
    (<
      (setq dcl_id
        (load_dialog "maindlg.dcl")
      )
    0)
    (exit)
  )
  (while (< 1 what_next)                    Maradjon a hurokban az OK-ig
    (new_dialog "maindlg" dcl_id)           vagy Mégsem
    (action_tile "x" "(subdlg)")            Akciók beállítása és set_tiles
    (cond
      ((= what_next1 3)                     Csak akkor igaz, ha pontkijelölésből tér
                                            vissza
        (subdlg)                            A SUBDLG rutin hívása
        (if (/= 3 what_next1)               A főpárbeszéd inicializálása
          (setq what_next
            (start_dialog)
          )
        )
      )
      (t (setq what_next
        (start_dialog))                     Végrehajtja az inicializálást
      )
    )
    (cond
      ((= 2what_next)                       Pontkijelölés
        (setq xx
          (getpoint "\nJelölje ki a pontot: ")
        )
      )
    )
  )
)
(defun subdlg()
  (new_dialog "subdlg" dcl_id)
  (action_tile "hide_all"
    "(done_dialog 3)")                      Alpárbeszéd elrejtése
  (setq what_next1 (start_dialog))
  (if (= 3 what_next1)
    (done_dialog 2)                         Főpárbeszéd elrejtése
  )
)
```

Figyelmeztetés Amikor több **new_dialog** hívással jelenít meg beágyazott párbeszédpaneleket, ügyelnie kell arra, hogy minden egyes **new_dialog** hívásnak legyen **done_dialog** párja (akár egy

visszajelzésből történik a hívása, akár nem). Ellenkező esetben az alkalmazás kiszámíthatatlan módon viselkedhet.

A **term_dialog** függvény egyszerre lezárja az összes párbeszédpanelt, nem ad azonban vissza státuszkódot, így az alkalmazásnak nincs módja arra, hogy különbséget tegyen egy beágyazott párbeszédpanel elrejtése és a párbeszédpaneleknek egy esetleges hiba miatti elvetése között.

Listaablakok és legördülő listák

Az AutoLISP a következő függvényeket biztosítja a listaablakok és legördülő listák kezelésére:

add_list end_list start_list

A listaablakban vagy legördülő listában megjelenő listát három függvény egymást követő hívásával állíthatja össze. Az összeállítást követően módosíthatja is a listát. Három lehetséges műveletet végezhet, melyeket a **start_list** függvény *operation* argumentuma határoz meg (lehetséges értékei zárójelben láthatók):

- Új lista létrehozása (3).

  A **start_list** hívását követően az **add_list** függvényt egymás után többször is meghívhatja. Minden egyes **add_list** hívás egy új tételt ad a listához. A lista kezelését az **end_list** hívásával fejezze be.

- Egy tétel módosítása a listában (1).

  A **start_list** hívását követően az **add_list** függvényt egyszer hívja meg, hogy kicserélje azt a tételt, amelynek indexét a **start_list** hívásban megadta. (Ha többször hívja meg az **add_list** függvényt, akkor mindig ugyanazt a tételt cseréli ki.) A lista kezelését az **end_list** hívásával fejezze be. A lista első tétele 0 indexű.

- Új tétel hozzáfűzése a listához (2).

  A **start_list** hívását követően hívja meg az **add_list** függvényt, hogy hozzáfűzzön egy tételt a listához. Amennyiben további **add_list** hívásokat hajt végre, több tétel adódik hozzá a listához egészen addig, amíg az **end_list** függvényt meg nem hívja.

Tekintet nélkül arra, hogy melyik listaműveletet hajtja végre, e három függvényt ebben a sorrendben kell meghívnia: **start_list**, utána **add_list** (esetleg többször), majd **end_list**.

A **mapcar** függvény különösen jól használható "nyers" AutoLISP listának egy listaablakban megjeleníthető listává történő átalakítására.

Tételezzük fel, hogy az `appnames` lista karakterláncokat tartalmaz, melyeket a "választék" nevű listaablakban szeretne megjeleníteni. A következő programrészlet összeállítja és megjeleníti a listát:

```
(start_list "választék")          A listaablak nevének megadása
(mapcar ' add_list appnames)      Az AutoLISP lista megadása
(end_list)
```

Mivel az alapértelmezés szerinti argumentum a lista létrehozásának megfelelő 3-as, a példában ez nem került megadásra.

A `list_box` zóna értéke az aktuálisan kiválasztott tétel (tételei) indexe (indexei), ezért amennyiben a programban szükség van egy indexhez tartozó aktuális szöveg ismeretére, akkor tárolni kell az eredeti listát. Ugyancsak nyomon kell követni minden olyan változtatást, amely az alábbi példákban bemutatott módszerekkel történik.

Új tételek hozzáfűzése a listához hasonló egy új lista létrehozásához. Tételezzük fel, hogy az alknevek lista 12 tételt tartalmaz, és Ön egy másik, `newnames` nevű listát szeretne hozzáfűzni:

```
(start_list "választék" 2)
(mapcar 'add_list newnames)
(end_list)
```

Mivel egy lista hozzáfűzése nem alapértelmezett, ezért a listaműveletet (2-es) meg kell adni.

Egyetlen tétel megváltoztatásához csak egy **`add_list`** hívásra van szükség. Ezúttal a megváltoztatni kívánt tétel indexét is meg kell adnia:

```
(start_list "választék" 1 5)  A lista hatodik tételét módosítja
(add_list "MEGLEPETÉS!")      Ne felejtse el, hogy az első index nulla
(end_list)
```

Megjegyzés Egy tételt csak úgy törölhet a listából, vagy szúrhat be oda, ha az egész listát elölről kezdve újra felépíti.

Egy `list_box` zóna értéke tartalmazhat bevezető szóközöket, különösen, ha Ön több tételt olvas ki egyszerre; ezért ne vesse alá karakterlánc-összehasonlításnak. Az értéket először alakítsa át egész számmá az **`atoi`** függvény segítségével, még a listaablak működése előtt. Használhatja a **`read`** függvényt is, amely egy szimbólumot automatikusan egész számmá alakít át. Feltételezve, hogy a justone nevű lista csak egyetlen kiválasztást enged meg, az alábbi programrészlet azt vizsgálja meg, hogy a harmadik tétel ki van-e választva. Először ellenőrizni kell, hogy a karakterlánc nem üres-e, ugyanis az **(atoi)**, illetve **`atoi`** függvények egyaránt 0-át adnak vissza az üres karakterlánc és a "0" esetében.

```
(setq index ( get_tile "justone"))
(cond
  ((/= index "")
```

```
        (= 2 (atoi index))
        ; A harmadik tétel feldolgozása
        ...
    )
)
```

Megjegyzés Egy legördülő lista értéke sohasem kezdődik szóközzel, ezért azt nem szükséges átalakítani (ártani természetesen nem árt). A legördülő listák nem engednek meg többes kiválasztást.

Amennyiben egy listaablak támogatja a többszörös kiválasztást, a programnak kell elvégeznie az átalakítást úgy, hogy végiglépked az érték-karakterlánc többszörös értékein. Az **MK_LIST** következő definíciója egy olyan listát ad vissza, amely az eredeti displist-nek csak a felhasználó által kiválasztott tételeit tartalmazza. (Ebben a példában a displist — a képernyőn megjelenített lista — globális változóként szerepel.) Az **MK_LIST** függvény hívásakor a listaablak aktuális `$value` értékét várja argumentumként.

```
(defun MK_LIST (readlist / count item retlist)
  (setq count 1)
  (while (setq item (read readlist))
     (setq retlist (cons (nth item displist) retlist))
     (while (and (/= " " (substr readlist count 1))
       (/= "" (substr readlist count 1)))
      (setq count (1+ count))
     )
     (setq readlist (substr readlist count))
  )
  (reverse retlist)
)
```

Mindkét fenti program működik abban a szélsőséges esetben is, amikor csak egy tétel van kiválasztva.

Ábrazónák és nyomógombok

Az AutoLISP a következő függvényeket biztosítja az ábrazónák és nyomógombok kezelésére

dimx_tile	end_image	slide_image	vector_image
dimy_tile	fill_image	start_image	

A függvényeket bemutató példákat ebben a részben, illetve a 13. fejezetben találja.

Ábrák létrehozása

Az ábrazónák és ábra-nyomógombok megszerkesztésére szolgáló függvényhívások sorozata hasonló a listakezelő függvényekéhez. A **sstart_image** függvény indítja el egy ábra megszerkesztését, és az **end_image** fejezi be. Argumentumok helyett azonban különálló függvényekkel határozhatja meg, hogy mi kerüljön az ábrába:

vector_image Egy vektort (egyetlen egyenes vonalat) rajzol az ábrába.

fill_image Egy kitöltött téglalapot rajzol az ábrába.

slide_image Egy AutoCAD diát rajzol az ábrába.

A vektorok és a kitöltött téglalapok jól használhatók egyszerű ábrák, például olyan színminták (kitöltött téglalapok) céljára, amilyeneket az AutoCAD Szín kiválasztása párbeszédpanelje alkalmaz a felhasználó által kiválasztott szín bemutatására. Bonyolultabb ábrákhoz a diák megfelelőbbek, megjelenítésük azonban időigényes lehet. Ha diákat használ, alkalmazza a lehető legegyszerűbbeket.

Megjegyzés Ha az ábrazónákban kitöltött objektumokkal (széles vonalláncok, testek, háromdimenziós felületek) rendelkező diákat használ, az ábrák csak körvonalként fognak megjelenni, kivéve ha azokat a ÁRNYAL paranccsal létrehozott ábrából készítette.

Mindhárom ábrakészítő függvény, a **vector_image**, a **fill_image** és a **slide_image** is, abszolút koordináták megadását kívánja meg. Ahhoz, hogy ennek maradéktalanul eleget tehessen, ismernie kell az ábrazóna vagy ábra-nyomógomb pontos méreteit. Mivel ezek a méretek általában a párbeszédpanel elrendezésekor alakulnak ki, a PDB csomag olyan függvényeket is biztosít, amelyek egy adott zóna szélességét és magasságát adják vissza. Az AutoLISP két ilyen méretfüggvényt kínál, a **dimx_tile** és a **dimx_tile** függvényt. Hívja meg ezeket, mielőtt hozzákezdene egy ábra létrehozásához. Az origó (0,0) mindig a zóna bal felső sarka.

A színek az AutoCAD színszámokkal vagy az alábbi táblázatban feltüntetett "logikai" színszámokkal adhatók meg. (Az értékeket és a mnemonikokat az Autodesk eszközfelület — Autodesk Device Interface, ADI® — definiálja.)

A szín attribútum szimbolikus nevei

Színszám	*ADI mnemonik*	*Jelentés*
–2	BGLCOLOR	A háttér aktuális színe az AutoCAD grafikus képernyőjén
–15	DBGLCOLOR	A háttér aktuális színe a párbeszédpanelben

A szín attribútum szimbolikus nevei (*folytatás*)

Színszám	*ADI mnemonik*	*Jelentés*
–16	DFGLCOLOR	Az előtér aktuális színe a párbeszédpanelben (szövegek számára)
–18	LINELCOLOR	A vonalak aktuális színe a párbeszédpanelben

A következő példában a "cur_color" egy ábrazóna, amelyet teljesen ki szeretne tölteni egy vörös folttal:

```
cur_color

(setq widht (dimx_tile  "cur_color")
      height (dimy_tile "cur_color"))
(start_image "cur_color")
(fill_image 0 0 widht height 1)     1= AutoCAD vörös
(end_image)
```

Az ábrakészítő függvények együttesen is használhatók. Az alábbi programrészlet kitölt egy ábrát, majd egy csíkkal áthúzza azt:

```
(setq widht (dimx_tile "csík")
      height (dimy_tile "csík"))
(start_image "csík")
(fill_image 0 0 widht height 3)  3 = AutoCAD zöld
(setq x (/ width 2))                  A vektor függőlegesen középre
                                      kerül
(vector_image x 0 x height 4)     4 = AutoCAD világoskék
(end_image)
```

A **slide_image** függvénnyel megjelenített diák lehetnek különálló (*.sld*) diafájlok, vagy egy diakönyvtár (*.slb*) részei is. A dia nevét ugyanúgy adja meg, ahogyan az AutoCAD BEDIA parancsa vagy egy AutoCAD menüfájl (*.mnu*) esetében. Azaz ha a dia egy *.sld* fájlban található, adja meg a nevét egyszerűen az *.sld* kiterjesztés nélkül (például "frntview"). Amennyiben a dia egy diakönyvtárban található, akkor adja meg előbb a könyvtár nevét (kiterjesztés nélkül), utána pedig magának a diának a nevét (szintén kiterjesztés nélkül) zárójelben (például "allviews(frntview)"). A **slide_image** függvény a diát vagy a diakönyvtárat az AutoCAD aktuális könyvtárkeresési útvonalán keresi (tekintse meg a "load_dialog" függvényt a 13. fejezetben).

Tételezzük fel, hogy a dia egy egyszerű fájl, neve *felulnez.sld*:

```
(setq x (dimx_tile "view")
      y (dimy_tile "view"))
(start_image "view")
( slide_image 0 0 x y "topview")
(end_image)
```

Diákban a vektorokat gyakran fehér színnel (7-es érték) rajzolják, ami egy ábra alapértelmezett háttérszíne. Ha az Ön ábrája üres, mialatt először egy diát jelenít meg, próbálja meg a `color` attribútumot `graphics_background` értékűre állítani. (A háttér színét úgy is átállíthatja, ha a **`slide_image`** előtt a **`fill_image`** függvényt is meghívja.)

Ábra-nyomógombok kezelése

Az ábra-nyomógombok ugyanúgy kezelhetők, mint a közönséges nyomógombok; vagyis amikor a felhasználó megnyomja, azzal egy műveletet indít el. A PDB szolgáltatás azonban régiók definiálásának a lehetőségét is biztosítja a nyomógombon, így a végrehajtandó akció annak függvénye lehet, hogy a felhasználó a nyomógomb *melyik részét* választotta ki. Ennek módszere egyszerű: az ábra-nyomógombhoz tartozó akció vagy visszajelzés visszaadja a felhasználó által kijelölt pont (*X*,*Y*) helyét. A koordináták az adott ábra-nyomógomb határain belülre esnek (ezek a méretfüggvényekkel határozhatók meg). Az alkalmazáson múlik, hogy a kijelölt pont helyének milyen jelentést tulajdonít; ily módon akár régiókat is megkülönböztethet az ábrában. A DDVPOINT párbeszéd ezt a tulajdonságot használja ki.

Tegyük fel, hogy a **`fill_image`** felhasználásával olyan ábra-nyomógombot készített, amelyben két különböző színárnyalatú terület található. Aszerint szeretné egyiket vagy másikat kiválasztani, hogy a felhasználó melyik területre mutatott rá. Amennyiben az ábra-nyomógomb vízszintesen van kettéválasztva (felül sötét, alul világos), akkor az akciónak elég csak az egyik méretet megvizsgálnia:

```
(action_tile "image_sel" "(pick_shade $key $value $y)")
  ...
(defun pick_shade (key val y)
  (setq threshold
    (/ ( dimy_tile  key) 2)) Az ábra vízszintesen
                              kettéválasztott
  (if (> y threshold)         Az origó a bal felső sarok
    (setq result "Világos")
  (setq result "Sötét") )
)
```

Alkalmazás-specifikus adatok

Az AutoLISP **`client_data_tile`** függvénye alkalmazás-specifikus adatot rendel egy zónához. Ez az adat visszajelzéskor a $data változóból olvasható ki, és mindig karakterlánc. A kliensadat nem szerepel a DCL fájlban; csak az alkalmazás futása idején érvényes. Használata a felhasználó által definiált attribútumokhoz hasonlítható. A legfőbb különbség az, hogy a felhasználó által definiált

attribútumok csak olvashatók, a kliensadat viszont változhat futásidő alatt. Ezenkívül, a végső felhasználók megszemlélhetik a felhasználó által definiált attribútumokat az alkalmazáshoz tartozó DCL fájlban, a kliensadat viszont láthatatlan a számukra.

Mivel egy listaablakban (vagy legördülő listában) megjelenő listát az Ön által írt programnak kell karbantartania, a kliensadat jó megoldás lehet ennek az információnak a kezelésére. Az **MK_LIST** függvény definíciójának alábbi módosításában maga a lista már argumentumként szerepel (az egész függvény a "Listaablakok és legördülő listák" című résznél, a 317. oldalon látható):

```
(defun MK_LIST (readlist displist / )
```

Ez szükségtelenné teszi a listának egy globális változóban történő tárolását. A párbeszédpanelt kezelő program fő részében található alábbi függvényhívások a **client_data_tile** hívásával egy rövid listát társítanak a zónához, azután ezt a listát egy akciókifejezés útján átadják az **MK_LIST** függvénynek:

```
(client_data_tile
  "színrendszlist"
  "Vörös-Zöld-Kék-Világoskék-Bíbor-Sárga-Színárnyalat-Telítettség-
Érték"
)
(action_tile
  "színrendszlist"
  "(setq usrchoice (mk_list $value $data))"
)
```

A párbeszédpanel-függvények összefoglalása

Ez a rész áttekinti és összefoglalja a tipikus párbeszédpanel-kezelő függvényeket. Összegzi a fejezetben korábban tárgyalt fogalmakat. Egy mintaalkalmazást is bemutat, amihez, egy új fájl megtervezésekor és végrehajtásakor folyamodhat.

Függvényhívások sorrendje

A következő rész egy tipikus függvényhívási sorrendet mutat be.

1 Töltse be a DCL fájlt a `load_dialog` függvény meghívásával.

2 A `new_dialog` függvény hívásásval jelenítse meg a párbeszédpanelt.

   Mindenképpen ellenőrizze a `new_dialog` függvény visszatérési értékét. A `start_dialog` függvény hívása megjósolhatatlan eredményre vezet, amennyiben a `new_dialog` függvény hívása meghiúsul.

3 Inicializálja a párbeszédpanelt a zóna értékeinek, valamint a listák és az ábrák beállításával. Ezt akkor is végezze el, amikor az `action_tile` függvényt hívja meg az akciókifejezések és a visszajelzési függvények beállítására. Más, ilyenkor szokásos függvényhívások a `set_tile` és `mode_tile`, melyek általános zónaértékeket és állapotokat állítanak be. A `start_list`, `add_list`, és `end_list` függvények a listaablakokat, a méretfüggvények a `start_image`, `vector_image`, `fill_image`, `slide_image`, és `end_image` függvényekkel az ábrákat kezelik. Ilyenkor a `client_data_tile` függvényt is meghívhatja, hogy egyeztesse az alkalmazásra vonatkozó adatokat a párbeszédpanellel és annak részeivel.

4 Hívja meg a `start_dialog` függvényt, hogy az irányítást átadja a párbeszédpanelnek, s így a felhasználó bemeneti adatokat adhasson meg.

5 Használja a felhasználói bevitelt az akcióból. Ez leginkább a `get_tile`, `get_attr`, `set_tile` és `mode_tile` függvények alkalmazásakor lehet hasznos.

6 A felhasználó megnyomja a kilépési billentyűt, hogy meghívja a `done_dialog` függvényt, ami a `start_dialog` függvény visszatérési értékét adja meg. Ekkor törölje a DCL fájlt a `unload_dialog` függvény meghívásával.

Ez a minta egyetlen párbeszédpanelt és egyetlen DCL fájlt kezel egyszerre. Az alkalmazásoknak általában több párbeszédpaneljük van. Ezen párbeszédpanelek használatának legkönnyebb és leggyorsabb módja, ha egy egyszerű DCL fájlban elmenti őket. A `load_dialog` függvény meghívja, majd egyszerre betölti az összes párbeszédpanelt.

A **new_dialog** függvényt bármely párbeszédpanel esetén meghívhatja. Amennyiben kevés a memória, szüksége lehet több DCL fájl létrehozására, valamint az **unload_dialog** függvény használatára. Ez utóbbival párbeszédpaneleket törölhet ki a memóriából, mielőtt egy új készletet töltene be a **load_dialog** függvénnyel.

Párbeszédpanel-minta blokk definiálásához

A *bmake.lsp* minta-alkalmazásfájl és a hozzá kapcsolódó *bmake.dcl* fájl számos párbeszédablak-kezelési technikát mutat be . Ezek a fájlok a *sample* könyvtárban találhatók. A *bmake* alkalmazás lényegében egy interaktív felhasználói felület az AutoLISP **entmake** függvényéhez. Használhatja új blokkok definiálására és már létező blokkok neveinek megtekintésére. Következzék néhány, a *bmake* fájlban található módszer.

- Párbeszédpaneleket rejthet el, ha speciális állapotkódot definiál a **done_dialog** függvényhez és továbbítja azt a **start_dialog** függvénynek. Nézze meg a **C:BMAKE** fő ciklusát (a **load_dialog** és a **action_tile** függvények hívása után található).
- A kapcsolót használhatja egyéb zónák engedélyezésére vagy letiltásásra. Tekintse át a **DO_UNNAMED** függvény definícióját.
- Listát készíthet a listaablak számára. Lásd a **PAT_MATCH** és a **SORT** függvényeket.
- Megjelenítheti az AutoCAD súgó párbeszédpanelt. Lásd a **DO_HELP** függvényt.

A párbeszédpanel-technikák mellett a *bmake* a helyes tervezést is illusztrálja.

11

AUTOLISP GYAKORLÓ PÉLDA

Ez a gyakorló rész bemutatja, hogy miként bővíthető új paranccsal az AutoCAD rendszer. Elmagyarázza az AutoLISP működését és azt is bemutatja, hogyan hasznosíthatja azt saját alkalmazásaiban. A következőkben létrehozandó parancs a kertépítészettel kapcsolatos, de az alapelvek bármely alkalmazási területre érvényesek.

Ez a rész gyakorlott AutoCAD felhasználók számára készült. A szükséges feladatok végrehajtásához rendelkeznie kell egy ASCII fájlok létrehozására alkalmas szövegszerkesztővel.

A kitűzött cél

Az cél egy olyan új AutoCAD parancs kifejlesztése, amely egy kerti járdát rajzol, és ezt feltölti kör alakú beton burkolólapokkal. Az új parancs promptsorozata a következő lesz:

Parancs: **kj**
A járda kezdőpontja:*Jelölje ki a kezdőpontot!*
A járda végpontja: *Jelölje ki a végpontot!*
A járda félszélessége: *Adjon meg egy számot!*
A burkolólapok sugara: *Adjon meg egy számot!*
A burkolólapok távolsága: *Adjon meg egy számot!*

A *kezdőpont* és a *végpont* megadásával határozhatja meg a járda középvonalát. Ezt követően a járda félszélességét és a kör alakú burkolólapok sugarát végül, a burkolólapok közti távolságot kell megadnia. Ha a teljes szélesség helyett a félszélességet használja, a kezdőponttól egy gumivonal segítségével szemléletesebben határozható meg a járda szélessége.

Lássunk munkához!

Ezt az alkalmazást a leggyakrabban használt módszerrel, belülről kifelé (vagy alulról felfelé) építkezve fogjuk kialakítani. Az alkalmazásban gyakran fogunk szögeket használni. Az AutoLISP, a legtöbb programnyelvhez hasonlóan, radiánban határozza meg a szögeket. A szögek radiánban kifejezve 0 és 2 * π (PI) közötti intervallumban. helyezkednek el. Mivel a legtöbb felhasználó a szögeket fokokban számolja, így egy olyan *függvényt fogunk definiálni, amely a szögeket radiánban kifejezett* értékekké alakítja át. A szövegszerkesztővel hozzon létre egy *kj.lsp* nevű fájlt. Írja be a következő programot:

```
;  Fokok átváltása radiánra

(defun dtr (a)
  (* pi (/ a 180.0))
)
```

Nézzük meg, mit csinál ez a program. A **defun** függvény segítségével egy AutoLISP függvényt definiáltunk. A függvény a **dtr** nevet kapta (a *degrees to radians* — fokok radiánra rövidítéseként). Ennek a függvénynek egyetlen argumentuma van, az amely a szög értékét jelöli fokokban kifejezve. A kifejezés eredménye:

```
π * (a / 180.0)
```

Ezt a LISP jelölést úgy olvashatja, hogy " π-nek, az *a* és a 180.0 hányadosával való szorzata". A π az AutoLISP által előre definiált érték, azaz 3.14159.... A pontosvesszővel kezdődő sor *megjegyzés*; a sorban a pontosvesszőt követő szöveget az AutoLISP nem veszi figyelembe.

Mentse el a szövegfájlt lemezre, majd indítsa el az AutoCAD programot egy új rajzzal (a név ebben az esetben nem érdekes, mivel úgysem mentjük el a rajzot). Amikor a Parancs prompt megjelenik, töltse be a függvényt az alábbiak szerint:

Parancs: **(load "kj")**
DTR

Az AutoLISP betölti a függvényt, és visszajelzi annak nevét (DTR). (Feltételezzük, hogy a kj.*lsp* az AutoCAD keresési útvonalán található.) Ettől kezdve valahányszor azt olvassa, hogy *indítsa el az AutoCAD rendszert, és töltse be a programot,* az előbb leírt eljárást hajtsa végre.

Most próbálja ki a függvényt különböző értékekkel. A radián fenti definíciója alapján a nulla foknak egyenlőnek kell lennie a nulla radiánnal, tehát írja be a következő sort:

Parancs: **(dtr 0)**

Amennyiben a promptra kerek zárójellel kezdődő sort billentyűz be, akkor ezt az AutoCAD kiértékelésre át fogja adni az AutoLISP számára. Ebben az esetben a frissen definiált, és nulla argumentumú `dtr` függvényt értékeltetjük ki. A kiértékelés után az AutoCAD kiírja az eredményt, tehát erre a bemenetre az alábbi válasznak kell megjelennie:

0.0

Most próbálkozzon a 180 fokkal. Amennyiben az alábbiakat billentyűzi be:

Parancs: **(dtr 180)**

a következő választ kapja:

3.14159

Ez azt jelzi, hogy 180 fok egyenlő π radiánnal. Ha megvizsgálja a függvényt, láthatja, hogy éppen ezt definiáltuk.

Amikor idáig eljutott, lépjen ki az AutoCAD rendszerből, és térjen vissza a szövegszerkesztőhöz.

Adatbevitel

A "kerti járda" parancs meg fogja kérdezni a felhasználót, hogy hová rajzolja az utat, milyen széles legyen, mekkora legyen a burkolólapok mérete, és hogy ezeket milyen távolságban helyezze el egymástól. Az alábbiakban egy olyan függvényt fogunk definiálni, amely mindezt megkérdezi a felhasználótól, ezt követően pedig kiszámítja a parancs által a továbbiakban felhasználásra kerülő értékeket.

A szövegszerkesztő segítségével adja hozzá a *kj.lsp* szövegfájlhoz az alábbi sorokat .

```
; A kerti járdára vonatkozó információ

(defun gpuser ()
  (setq sp (getpoint "\nA járda kezdőpontja: "))
  (setq ep (getpoint "\nA járda végpontja: "))
  (setq hwidth (getdist "\nA járda félszélessége: " sp))
  (setq trad (getdist "\nA burkolólapok sugara: " sp))
  (setq tspac (getdist "\nA burkolólapok távolsága: " sp))

  (setq pangle (angle sp ep))
  (setq plength (distance sp ep))
  (setq width (* 2 hwidth))
  (setq angp90 (+ pangle (dtr 90))) ; a járda dőlése + 90
fok
  (setq angm90 (- pangle (dtr 90))) ; a járda dőlése - 90
fok
)
```

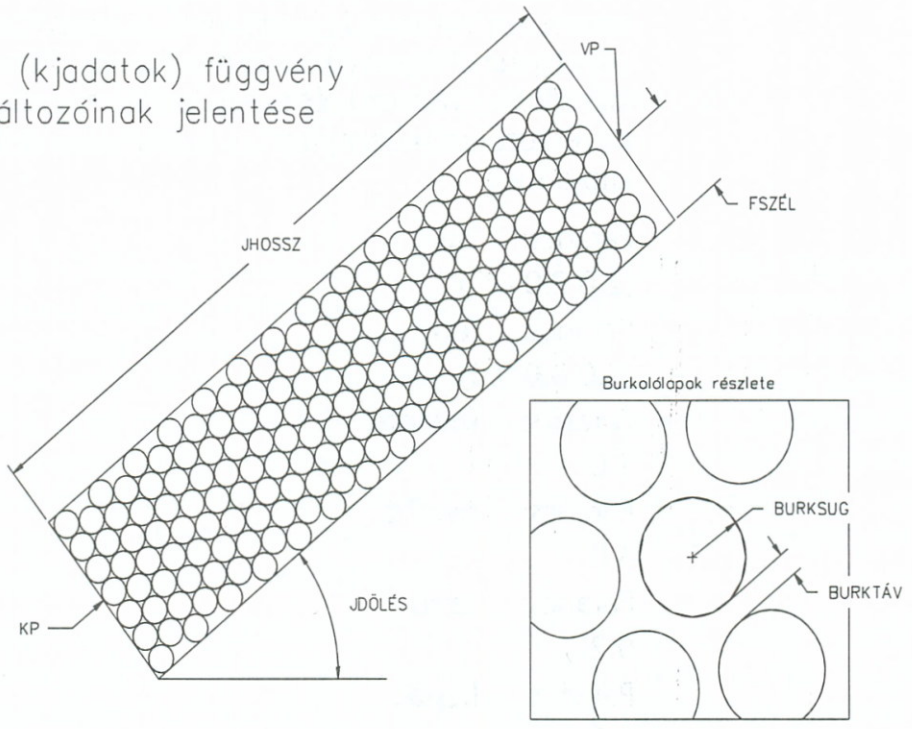
A program kifejezéseit nem szükséges tabulálni. Gyakorlatilag az egész programot egyetlen sorba is írhatta volna. A tabulálással és a sorok tördelésével azonban világosabbá és áttekinthetőbbé tehető a program szerkezete. A fontosabb kifejezések kezdő és befejező zárójeleinek egy vonalba rendezésével azt is biztosíthatja, hogy ne legyenek hiányzó, vagy felesleges zárójelek. Azt ajánljuk, hogy a sorok igazításához tabulátorok helyett használjon szóközöket. Ezzel biztosíthatja, hogy a tördelés változatlan marad a szerkesztési alkalmak során, és független lesz az adott szövegszerkesztőtől.

A fentiekben egy **gpuser** nevű függvényt definiáltunk. Ennek a függvénynek nincsenek argumentumai, csak megkérdezi a felhasználótól a kívánt adatokat. A **setq** függvény egy AutoLISP változót állít be a meghatározott értékre. Az első **setq** a **getpoint** függvény eredményét adja át az sp (start point) változónak. A **getpoint** függvény kérni fogja a felhasználót, hogy adjon meg egy pontot. Az AutoCAD az ezt követő karakterláncot fogja promptként használni. A \n vezérlő karakter hatására a prompt új sorban fog megjelenni. A **getdist** függvénnyel a járda félszélességét, a burkolólap átmérőjét, és a lapok közötti távolságot kapjuk meg. A **getdist** második argumentuma az sp, amely a távolság *kiindulópontját* határozza meg. A távolság ebből úgy adódik, hogy az

AutoCAD segítségével megad egy pontot, és a kezdőpont és ezen pont közötti távolság (amelyet az AutoCAD egy kifeszített gumivonalként jelenít meg) határozza majd meg a kért távolságot.

A felhasználói adatbevitelt követően a program kiszámít néhány gyakran használt változót. A `pangle` változó a járda kezdő- és végpontja közötti szög értékét kapja meg. Az **`angle`** függvény számítja ki a két pontot összekötő egyenes dőlésszögét. A `plength` változó értéke a járda hosszával lesz egyenlő. A **`distance`** függvény számítja ki a megadott két pont közötti távolságot. Mivel a járdának csak a félszélességét kellett megadni, a teljes szélesség ennek kétszerese. Végül kiszámítja és elmenti a járda 90 fokkal megnövelt, illetve lecsökkentett dőlésszögét az `angp90`, illetve az `angm90` változókban (mivel a szögeket az AutoLISP radiánban fejezi ki, a számítások előtt a szögek radiánná konvertálásához a **dtr** függvényt kell használnia).

Az alábbi ábra azt mutatja, hogy a **kjadatok** függvénynek átadott változók hogyan határozzák meg a járda méreteit.



Mentse el az átdolgozott programot a lemezre, indítsa el az AutoCAD rendszert, és töltse be a programot. Próbálja ki az adatok megadását kérő függvényt, és győződjön meg helyes működéséről. A függvényt a következőképpen aktivizálhatja:

Parancs: **(gpuser)**

Válaszolja a következőket a promptokhoz:

A járda kezdőpontja: **2,2**
A járda végpontja: **9,8**
A járda félszélessége: **2**
A burkolólapok sugara: **.2**
A burkolólapok távolsága: **.1**

A **gpuser** függvény a válaszokból kiszámítja a számára szükséges további változókat, és megjeleníti az utolsó számítás eredményét (ebben az esetben a -0.86217 értéket, azaz az `angm90` változó értékét radiánban). A **gpuser** függvénnyel beállított összes változó értékét tetszés szerint meg is jelenítheti, amennyiben egy felkiáltójelet (!) követően bebillentyűzi a változó nevét. Ebben az esetben az AutoCAD kiértékeli a változót, és megjeleníti az eredményt. Amennyiben a következőket billentyűzi be, akkor a zárójelben levő eredményeket kapja meg:

Parancs: **!sp**
(2.0 2.0 0.0)
Parancs: **!ep**
(9.0 8.0 0.0)
Parancs: **!hwidht**
2.0
Parancs: **!widht**
4.0
Parancs: **!trad**
0.2
Parancs: **!tspac**
0.1
Parancs: **!pangle**
0.708626
Parancs: **!plenght**
9.21954
Parancs: **!angp90**
2.27942
Parancs: **!angm90**
-0.86217

Az `sp` és az `ep` változók értéke háromdimenziós pontként (*X*,*Y* és *Z* koordinátákkal) jelenik meg, bár ebben a gyakorlatban eltekintünk a *Z* összetevőtől.

A `pangle`, az `angp90` és az `angm90` értéke radiánban kifejezve jelenik meg. Miután ezeket az értékeket ellenőrizte, lépjen ki az AutoCAD rendszerből, és indítsa el a szövegszerkesztőt a *kj.lsp* fájl szerkesztéséhez.

Tájékozódás

A felhasználó által megadott helyzettel már megrajzolható a járda. A függőleges vonallal megjelölt sorokat illessze be a *kj.lsp* fájlba:

```
; Fokok átváltása radiánra

(defun drawout ()
  (command "pline"
    (setq p (polar sp angm90 hwidth))
    (setq p (polar p pangle plength))
    (setq p (polar p angp90 width))
    (polar p (+ pangle (dtr 180)) plength)
    "close"
  )
)
```

Ez a bővítés egy **`drawout`** nevű függvényt definiál. Ez a függvény a **`gpuser`** függvényből kapott kezdőpont, dőlésszög, valamint a járda hosszúsági adatainak segítségével Vonalláncként rajzolja meg a járda körvonalát. A **`drawout`** függvény a **command** függvényt használja arra, hogy parancsokat és adatokat adjon át az AutoCAD rendszernek. A **`command`** függvény segítségével az AutoLISP függvényei parancsok végrehajtására utasítják az AutoCAD rendszert. A **`command`** függvénynek tetszőleges számú argumentuma lehet; ezeket egyenként átadja az AutoCAD rendszernek.

Ebben az esetben az AutoCAD a VLÁNC parancsot, majd a járda négy sarokpontját kapja meg. Az egyes sarokpontokat a **`polar`** függvény segítségével számítjuk ki, és az ideiglenes p változóban tároljuk. A **`polar`** függvény első argumentuma egy pont, a második és a harmadik pedig egy szög és egy távolság. A függvény eredményeként az eredeti ponttól a megadott távolságra és a megadott szögben lévő pontot kapjuk meg. Jelen esetben a járda kezdőpontjából kiindulva a járda négy geometriai határolópontját határozzuk meg. A parancs lezárásához a *zár* karakterláncot továbbítjuk a VLÁNC parancsnak, amely ennek hatására megrajzolja a járda negyedik, lezáró oldalát, és visszatér az AutoCAD Parancs promptjához.

A függvény kipróbálásához mentse el az átszerkesztett *kj.lsp* szövegfájlt, indítsa el az AutoCAD rendszert egy új rajzzal, és az előzőekhez hasonlóan töltse be az AutoLISP fájlt. Az előzőekhez hasonlóan aktivizálja az adatbevitelt kérő függvényt:

Parancs: **(gpuser)**

Adja meg az értékeket, mint az előző lépésben tette. Ezt követően próbálja ki az új **drawout** függvényt:

Parancs: **(drawout)**

A függvény átadja a járda körvonalának megrajzolására vonatkozó parancsokat az AutoCAD rendszernek, és a körvonal megjelenik a képernyőn. A függvény kipróbálása után lépjen ki az AutoCAD rendszerből.

A burkolólapok megrajzolása

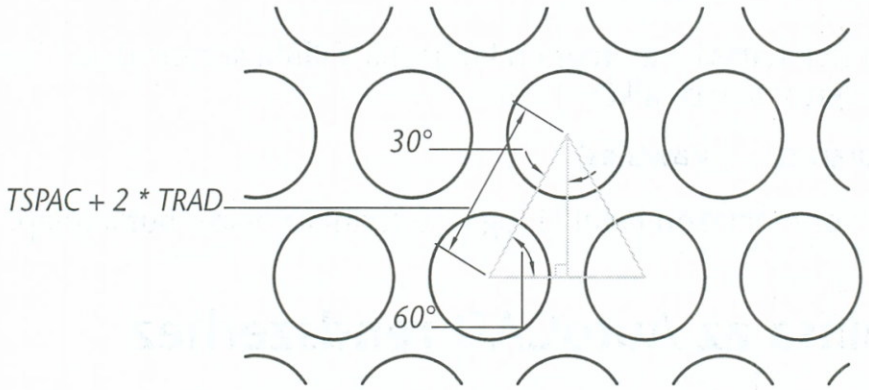
Most, hogy létrehozta és kipróbálta az adatokat bekérő és a körvonalat kirajzoló függvényeket, kitöltheti a járda körvonalát kör alakú burkolólapokkal. Ennek során geometriai ismeretekre is szüksége lesz. Indítsa el a szövegszerkesztőt, és illessze be a programba az alábbiakat:

```
; Egy sor burkolólap elhelyezése a járda hosszában,
; és esetleges párhuzamos eltolása

(defun drow (pd offset)
  (setq pfirst (polar sp pangle pd))
  (setq pctile (polar pfirst angp90 offset))
  (setq p1tile pctile)
  (while (< (distance pfirst p1tile) (- hwidth trad))
    (command "circle" p1tile trad)
    (setq p1tile
      (polar p1tile angp90 (+ tspac trad trad)))
  )
  (setq p1tile (polar pctile angm90 (+ tspac trad trad)))
  (while (< (distance pfirst p1tile) (- hwidth trad))
    (command "circle" p1tile trad)
    (setq p1tile
      (polar p1tile angm90 (+ tspac trad trad)))
  )
)
; A burkolólapok sorainak megrajzolása
```

```
(defun drawtiles ()
  (setq pdist (+ trad tspac))
  (setq off 0.0)
  (while (<= pdist (- plength trad))
    (drow pdist off)
    (setq pdist
      (+ pdist (* (+ tspac trad trad) (sin (dtr 60)))))
    (if (= off 0.0)
      (setq off (* (+ tspac trad trad) (cos (dtr 60))))
      (setq off 0.0)
    )
  )
)
```

Az alábbi ábra segít megérteni a függvényeket. A **drow** függvény egy sor burkolólapot rajzol ki a járda mentén az első argumentuma által megadott távolságra, és a járdára merőlegesen eltolva a második argumentummal megadott távolságra. Az esztétikusabb elrendezés, és a nagyobb terület lefedése érdekében a burkolólapokat elcsúsztatott sorokban helyezi el.



A **drow** függvény úgy keresi meg a sor első burkolólapjának helyét, hogy a **polar** függvény révén az első argumentummal megadott távolságra mozdul el a járda mentén, majd erre merőleges irányban. Ezt követően a **while** függvénnyel folytatja a körök rajzolását, amíg el nem ér a járda széléig. A **while** ciklus végén lévő **setq** függvény a következő burkolólapot két burkolólap-sugár plusz a lapok közötti távolságra helyezi el. A második **while** ciklus ezt követően az ellenkező irányban rajzolja a következő burkolólapsort, amíg el nem éri a járda szélét.

Az összes burkolólapsor megrajzolásához a **drawtiles** függvény újra és újra meghívja a **drow** függvényt. A **while** ciklusa hosszában végighalad a járdán, és minden lépésnél kirajzoltat egy sort a **drow** függvénnyel. A szomszédos sorokban levő lapok egyenlő oldalú háromszögeket alkotnak, ahogy azt az ábra is mutatja. A háromszögek éleinek hossza megegyezik a burkolólap-sugár kétszeresének és a lapok közötti távolságnak az összegével. Trigonometriai számításokkal a járda sorai közötti távolságot úgy kaphatja meg, ha ezt az értéket

megszorozza 60 fok szinuszával; a páratlan sorok közötti eltolás pedig az érték cos(60)-szorosa.

A **drawtiles** függvény **if** része a sorok eltolására szolgál. Az **if** függvény először megvizsgálja az első argumentumát. Amennyiben ez igaz, a második argumentumot hajtja végre, ellenkező esetben pedig a harmadikat. Esetünkben, ha az OFF egyenlő nullával, értékét az előzőekben ismertetett módon a burkolólapok távolságának cos(60)-szoros értékére kell állítani. Amennyiben az OFF változó értéke nem nulla, akkor állítsa nullára. Ezzel váltogatható a sorok tengelyirányú elcsúsztatása.

A függvény kipróbálásához mentse el a szövegfájlt, majd indítsa el az AutoCAD rendszert, és töltse be a programot. Billentyűzze be az alábbiakat:

Parancs: **(gpuser)**

Az előzőekhez hasonlóan adja meg a járda adatait. Ezt követően írja be az alábbiakat:

Parancs: **(drawout)**

Ennek hatására a program kirajzolja a járda körvonalait. Végül billentyűzze be a következőt:

Parancs: **(drawtiles)**

A körvonalakon belül ki fog rajzolódni az összes burkolólap.

A parancs csatolása az AutoCAD rendszerhez

Utolsó lépésben szerkessze össze a részeket egyetlen AutoCAD paranccsá. Amennyiben **c:xxx** név alatt definiál egy AutoLISP függvényt, az ***xxx*** név bebillentyűzése meg fogja hívni azt (feltételezve, hogy az ***xxx*** nem egy létező AutoCAD parancs neve). A KJ parancs teljessé tételéhez definiáljon egy ***c:KJ*** függvényt; ez lehetővé teszi, hogy a *kj.lsp* betöltése után a "kerti járda" parancs a kj bebillentyűzésével bármikor használható legyen.

A KJ parancs végrehajtása közben minden meghívott AutoCAD parancs megjelenik a parancssor területen, a kiválasztott pontokban pedig a pontkijelöléskor megszokott kis kereszt jelenik meg. Ha a parancsfüggvény már hibamentes, érdemes kikapcsolni ezeket a visszajelzéseket, hogy a függvény teljesen megegyezzen az AutoCAD beépített parancsaival.

A szövegszerkesztő segítségével illessze a megjelölt sorokat a *kj.lsp* szövegfájlhoz, majd indítsa el az AutoCAD rendszert, és töltse be a programot.

```
; A parancs végrehajtása részfunkcióinak meghívásával

(defun C:PATH ()
  (gpuser)
  (setq sblip (getvar "blipmode"))
  (setq scmde (getvar "cmdecho"))
  (setvar "blipmode" 0)
  (setvar "cmdecho" 0)
  (drawout)
  (drawtiles)
  (setvar "blipmode" sblip)
  (setvar "cmdecho" scmde)
  (princ)
)
```

A C:KJ függvény elkészítésével az AutoCAD egy KJ nevű paranccsal bővült.

A **getvar** függvény segítségével megkapjuk az AutoCAD BLIPMODE és CMDECHO rendszerváltozóinak aktuális értékét. A **setq** függvénnyel ezeket elmentjük az `sblip` illetve az `scmde` változókba. Ezután a **setvar** függvény segítségével mindkét AutoCAD változót nullára állítva letiltjuk a pontjelek és a parancsok megjelenítését. Ezeket a változókat csak az adatoknak a **gpuser** függvénnyel történő bekérése után állítjuk nullára, így a pontjelek a képernyőn maradnak, megerősítve ezzel a felhasználó által bevitt adatokat.

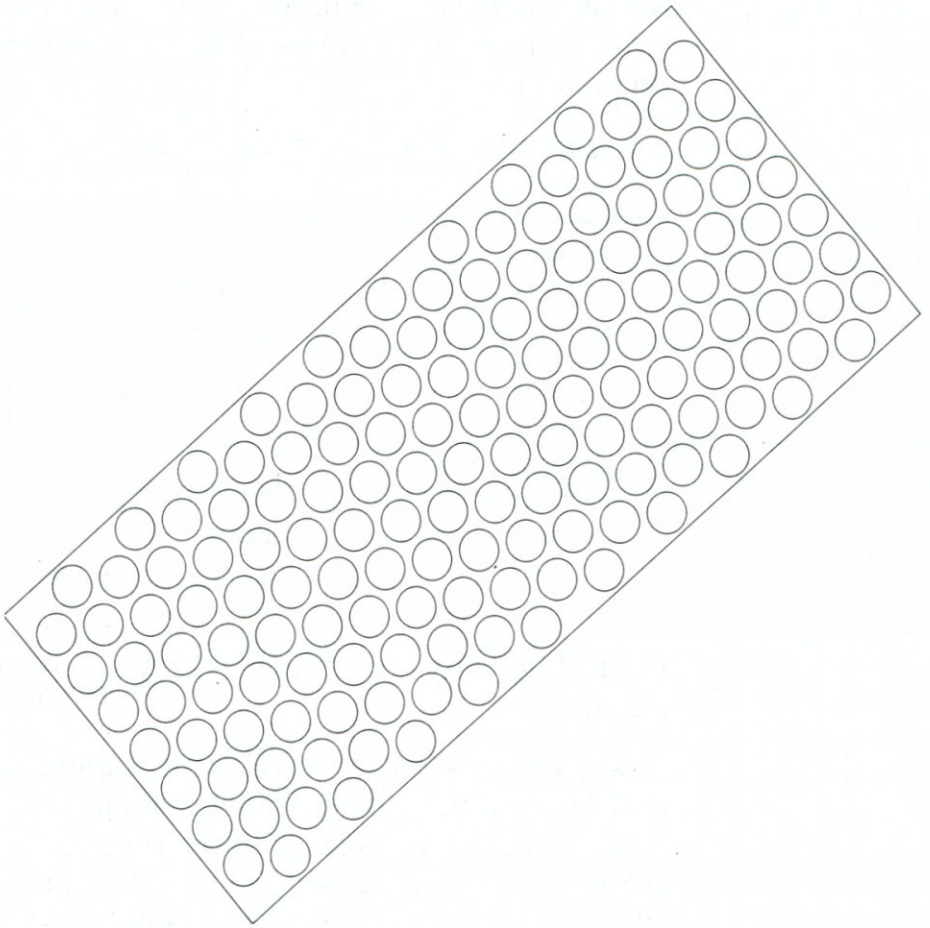
Miután a program befejezte a kerti járda kirajzolását, a **setvar** függvénnyel mindkét AutoCAD változót visszaállítjuk eredeti értékére.

Végül a **princ** függvény meghívásával elérhetjük, hogy a C:KJ függvény "*csendben*" fejeződjön be. Az AutoLISP függvények ugyanis mindig az utolsó függvényhívás eredményével térnek vissza. Ebben az esetben a **princ** függvény elhagyásával — 1 vagy 0 jelenik meg.

Mentse el a fájlt, indítsa el az AutoCAD rendszert, és próbálja ki a KJ parancsot a következő értékek megadásával.

Parancs: **kj**
A járda kezdőpontja: **2,2**
A járda végpontja: **9,8**
A járda félszélessége: **2**
A burkolólapok sugara: **.2**
A burkolólapok távolsága: **.1**

Ezzel az alábbi ábrának megfelelő kerti járdát kell kapnunk:



A kijelölő eszköz, illetve a billentyűzet használatával különböző bemeneteket meghatározva kísérletezhet a KJ paranccsal.

Kiegészítés párbeszédpaneles felülettel

AutoLISP programjaihoz a Párbeszédvezérlő nyelv (DCL) segítségével definiálhat párbeszédpaneleket.

A KJ parancs a parancssorban várja az adatokat a felhasználótól. A parancsot minden nehézség nélkül kiegészítheti azonban egy párbeszédes felülettel, ha elkészíti a párbeszédpanel leírását tartalmazó *.dcl* fájlt. A párbeszédpaneleket kezelő AutoLISP függvényekről és a DCL nyelvről ezen útmutató III. részében, valamint az AutoCAD párbeszédpanel-függvényei között, a 10. fejezetben a "Párbeszédpanelek kezelése" című részben olvashat.

A párbeszédpanelek nagyon jól használhatók arra, hogy a felhasználó választhasson bizonyos opciók közül. Mivel ez a program jelenleg nagyon kevés választási lehetőséget kínál, most további tulajdonságokkal fogjuk kiegészíteni.

A `C:KJ` függvény új szolgáltatása lehetővé teszi a burkolólapok alakjának meghatározását. Hibakezelő funkcióval is ellátjuk a függvényt.

Első lépésként készítsen másolatot a *kj.lsp* már elkészült változatáról *dpkj.lsp* névvel (az AutoCAD lemezeken található, párbeszédes felülettel rendelkező AutoLISP programok eredeti angol neve többnyire *ddxxx.lsp*, az általuk definiált parancsok magyar neve viszont DPXXX alakú). Ezen kívül egy *dpkj.dcl* nevű új ASCII fájl létrehozására is szükség lesz, amely a párbeszédpanel DCL nyelvű leírását tartalmazza.

A DCL fájl—*dpkj.dcl*

A párbeszédpanel, amelyet létre kell hoznia, két rádiógombot (ha az egyiket benyomja, a másik kiugrik) fog tartalmazni a burkolólap kétféle alakja (kör vagy poligon) közötti választásra. Három szerkesztőablak is lesz benne, melyekkel megadhatók a számszerű adatok: a burkolólap sugara, a burkolólapok közötti távolság, és a burkolólap éleinek száma (ez utóbbi csak akkor lesz elérhető, ha a sokszög rádiógombot választja ki).

A következő lépések szükségesek ahhoz, hogy a C:KJ függvényt egy egyszerű párbeszédpanellel egészítse ki. Az alábbiakat írja be a *ddkj.dcl* fájlba.

```
/*  DPKJ.DCL - DCL fájl DPKJ.LSP programhoz*/
gp_box1 : dialog {
     label = "A kerti járda burkolólapjainak meghatározása";
     : boxed_radio_row {
       label = "burkolólapok jellemzői";
       : radio_button {
         label = "&Sokszög";
         key = "gp_poly";
       }
       : radio_button {
         label = "&Kör";
         key = "gp_circ";
         value = "1";
       }
     }
     : edit_box {
       label = "&A burkolólap sugara";
       key = "gp_trad";
       edit_width = 6;
     }
     : edit_box {
       label = "&Burkolólapok távolsága";
       key = "gp_spac";
       edit_width = 6;
     }
     : edit_box {
       label = "&A burkolólap éleinek száma";
       key = "gp_side";
       edit_width = 4;
     }
```

A sokszög rádiógomb definíciója

A kör rádiógomb definíciója

A rádiógomb-terület definíciója

A Burkolólap sugara szerkesztőablak definíciója

A Burkolólapok távolsága szerkesztőablak definíciója

A Burkolólap éleinek száma szerkesztőablak definíciója

```
    : row {
      : spacer { width = 1; }
      : button {
        label = "OK";
        key = "accept";
        width = 8;
        fixed_width = true;
      }
      : button {
        label = "Mégsem";
        is_cancel = true;
        key = "cancel";
        width = 8;
        fixed_width = true;
      }
      : spacer { width = 1;}
    }
}
```

Az OK nyomógomb definíciója

A Mégsem nyomógomb definíciója

Az OK/Mégsem nyomógomb-sor

Párbeszédpaneleket kezelő AutoLISP függvények—*dpkj.lsp*

Miután elkészítette a DCL fájlt, következzék a *kj.lsp* fájlból létrehozott *dpkj.lsp* fájl továbbfejlesztése.

A függvények neve elárulja, hogy milyen műveletet hajtanak végre. A **load_dialog** függvény betölt egy párbeszédpanelt. A **set_tile** függvény egy adott zóna (tile) kezdeti értékét egy karakterlánc értékre állítja (a különböző nyomógombokat, szerkesztőablakokat, stb. zónáknak nevezzük), az **action_tile** függvény pedig meghatározza azt az akciót (műveletet), amely akkor fog lezajlani, miután a felhasználó aktivizálta az adott zónát.

A programba beillesztett új sorokat a **;<-ÚJ** megjegyzéssel különböztetjük meg. A régebbi sorok közül néhányat el kell távolítania a programból, akár úgy, hogy megjegyzéssé alakítja, akár úgy, hogy törli őket a szövegfájlból. Ezeknek a soroknak az elejére két pontosvesszőt (;;) tettünk, végükre pedig a ;<-ELTÁVOLÍT megjegyzést írtuk, és **kövér** betűtípussal szedtük őket.

```
;;;  DPKJ.LSP - a jó öreg kerti járda egy kicsit megbonyolítva.

; Fokok átváltása radiánra

(defun dtr (a)
 (* pi (/ a 180.0))
)
; A kerti járdára vonatkozó információ

(defun gpuser ()
  (setq sp (getpoint "\nA járda kezdőpontja: "))
  (setq ep (getpoint "\nA járda végpontja: "))
```

```
  (setq hwidth (getdist "\nA járda félszélessége: " sp))
;; (setq trad (getdist "\nA burkolólapok sugara: " sp))        ;<-
ELTÁVOLÍT
;; (setq tspac (getdist "\nA burkolólapok távolsága: " sp)  ;<-
ELTÁVOLÍT
(setq pangle (angle sp ep))
  (setq plength (distance sp ep))
  (setq width (* 2 hwidth))
  (setq angp90 (+ pangle (dtr 90)))
  (setq angm90 (- pangle (dtr 90)))
)

; A járda körvonalának megrajzolása

(defun drawout ()
  (command "vlánc"
        (setq p (polar sp angm90 hwidth))
        (setq p (polar p pangle plength))
        (setq p (polar p angp90 width))
        (polar p (+ pangle (dtr 180)) plength)
        "close"
  )
)
```

Írja be a következő sorokat a **drawout** függvény után.

```
; Párbeszédpanel hívása a burkolólapok adatainak beállításához


(defun gp_dialog ()
  (setq tshape "Kör"
       trad 0.5
       tspac 0.1
       tsides 8 )
  (setq dcl_id  (load_dialog "ddgp.dcl"))
  (if (not (new_dialog "gp_box1" dcl_id))(exit))
  (set_tile "gp_trad" "0.5")
  (set_tile "gp_spac" "0.1")
  (mode_tile "gp_side" 1)
  (set_tile "gp_side" "8")
  (action_tile "gp_circ"
    "(setq tshape \"Kör\")(mode_tile \"gp_side\" 1)")
  (action_tile "gp_poly"
    "(setq tshape \"Sokszög\")(mode_tile \"gp_side\" 0)")
  (action_tile "Mégsem" "(done_dialog)(setq gperr \"\")(exit)")
  (action_tile "accept"
    (strcat
      "(progn (setq trad (atof  (get_tile \"gp_trad\")))"
      "(setq tspac (atof (get_tile \"gp_spac\")))"
      "(setq tsides (atoi (get_tile \"gp_side\")))"
      " (done_dialog))"
    )
  )
```

```
  (start_dialog)
  (unload_dialog dcl_id)
  (if (= tshape "Circle")
    (defun gp_tile () (command "kör" p1tile trad))
    (defun gp_tile () (command "sokszög" tsides p1tile "" trad))
  )
)

; Hibakezelő definiálása

(defun gp_err (msg)
  (setq *error* olderr)
  (if (not gperr)
    (princ (strcat "\nKerti járda hiba: " msg))
    (princ gperr)
  )
  (if sblip (setvar "blipmode" sblip))
  (if scmde (setvar "cmdecho" scmde))
  (princ)
)
```

A következő program már megtalálható a *dpkj.lsp* fájlban. Módosítsa a jelzéseknek megfelelően.

```
; Egy sor burkolólap elhelyezése a járda hosszában,
; és esetleges párhuzamos eltolása

(defun drow (pd offset)
  (setvar "snapang" pangle)                          ;<-ÚJ
  (setq pfirst (polar sp pangle pd))
  (setq pctile (polar pfirst angp90 offset))
  (setq p1tile pctile)
  (while (< (distance pfirst p1tile) (- hwidth trad))
    (gp_tile)                                        ;<-ÚJ
;;  (command "kör" p1tile trad)                      ;<-ELTÁVOLÍT
    (setq p1tile (polar p1tile angp90 (+ tspac trad trad)))
  )
  (setq p1tile (polar pctile angm90 (+ tspac trad trad)))
  (while (< (distance pfirst p1tile) (- hwidth trad))
    (gp_tile)                                        ;<-ÚJ
;;  (command "kör" p1tile trad)                      ;<-ELTÁVOLÍT
    (setq p1tile (polar p1tile angm90 (+ tspac trad trad)))
  )
)
```

```
; A burkolólapok sorainak megrajzolása

(defun drawtiles ()
  (setq pdist (+ trad tspac))
  (setq off 0.0)
  (while (<= pdist (- plength trad))
    (drow pdist off)
    (setq pdist (+ pdist (* (+ tspac trad trad) (sin (dtr 60)))))
    (if (= off 0.0)
      (setq off (* (+ tspac trad trad) (cos (dtr 60))))
      (setq off 0.0)
    )
  )
)

; A parancs végrehajtása részfunkcióinak meghívásával

(defun C:DPJÁRDA ()                    ; <- MÓDOSÍTÁS, tegye a járdához
"DP"-t
  (setq olderr *error*                                    ; <- ÚJ
      *error* gp_err                                      ; <- ÚJ
      sblip nil                                           ; <- ÚJ
      scmde nil                                           ; <- ÚJ
      gperr nil                                           ; <- ÚJ
  )                                                       ; <- ÚJ
  (gpuser)
  (setq sblip (getvar "blipmode"))
  (setq scmde (getvar "cmdecho"))
  (setq sang (getvar "snapang"))                          ; <- ÚJ
  (setvar "blipmode" 0)
  (setvar "cmdecho" 0)
  (drawout)
  (gp_dialog)                                             ; <- ÚJ
  (drawtiles)
  (setvar "blipmode" sblip)
  (setvar "cmdecho" scmde)
  (setvar "snapang" sang)                                 ; <- ÚJ
  (setq *error* olderr)                                   ; <- ÚJ
  (princ)
)

; Print message once loaded.                              ; <- ÚJ

(princ "\nDPKJ.LSP betöltve. DPKJ beírásával használható..") ; <- ÚJ
(princ)                                                   ; <- ÚJ
```

12

AZ AUTOLISP FÜGGVÉNYEK ÁTTEKINTÉSE

E fejezet listái segítenek megtalálni egy ismeretlen nevű függvényt. Minden AutoLISP függvényt egy rövid leírás követ. A függvényeket működésük szerint az alábbi csoportokba rendeztük:

- *Alapfüggvények* (aritmetikai, karakterlánc-kezelő, egyenlőségi és feltételes, listakezelő, szimbólumkezelő, függvénykezelő, hibakezelő és alkalmazáskezelő függvények)
- *Segédfüggvények* (lekérdező- és parancsfüggvények, megjelenítésvezérlő, felhasználói bevitelre szolgáló, geometriai, konverziós, fájlkezelő és eszközelérési függvények)
- *Az ASE felület függvényei* (ASE adminisztratív, csatoláselérési, csatolás kiválasztás-kezelő, segéd- és hibakezelő függvények)
- *Kiválasztási halmaz, objektum és szimbólumtábla függvények* (kiválasztási halmazokat, objektumokat,bővített adatokat és szimbólumtáblákat kezelő függvények)
- *Programozható párbeszédpanel függvények* (párbeszédpanel nyitó és záró, illetve zónákat és attribútumokat, listaablakokat és legördülő listákat, ábrazónákat és alkalmazás-specifikus adatokat kezelő függvények)
- *Memóriakezelő függvények*

A függvényeket adattípus és működés szerint csoportosítottuk. A következő információk megfelelnek a 7., 8., 9. és 10. fejezetben leírtaknak. Részletes információ található az AutoLISP függvényekről a 13., "AutoLISP függvények katalógusa" című fejezet ábécé-sorrendbe szedett listájában.

Alapfüggvények

Az alapfüggvények aritmetikai, karakterlánc-kezelő, egyenlőségi és feltételes, listakezelő, szimbólumkezelő, függvénykezelő, hibakezelő alkalmazáskezelő függvényekből állnak. Ezeket részletesebben a 7., "Az AutoLISP alapjai" című fejezet tárgyalja.

Aritmetikai függvények

Az AutoLISP aritmetikai függvényei a következők:

(+ *[number number] ...*)

A számok összegével tér vissza.

(- *[number number] ...*)

Az első számból kivonja a következő számok összegét, és a különbséggel tér vissza.

(* *[number number] ...*)

A számok szorzatával tér vissza.

(/ *[number number] ...*)

Az első számot elosztja a többi szám szorzatával, és a hányadossal tér vissza.

(~ *int*)

Az argumentumon bitenként elvégzett NOT (egyes komplemens) művelet értékével tér vissza.

(1+ *number*)

Az argumentum eggyel megnövelt értékével tér vissza.

(1- *number*)

Az argumentum eggyel csökkentett értékével tér vissza.

(abs *number*)

Az argumentum abszolút értékével tér vissza.

(atan *num1 [num2]*)

Egy szám árkusz tangensével tér vissza radiánban.

(cos *angle*)

Egy radiánban kifejezett szög koszinuszával tér vissza.

(exp *number*)

A konstans e (egy valós szám) adott hatványával tér vissza (a természetes alapú logaritmus ellentetje).

(expt *base power*)

Egy szám adott hatványával tér vissza.

(fix *number*)

Egy valós számot a hozzá legközelebbi, kisebb egésszé konvertál.

(float *number*)

Egy szám valós számmá konvertált értékével tér vissza.

(gcd *int1 int2*)

Visszaadja két egész szám legnagyobb közös osztóját.

(log *number*)

Egy szám természetes alapú logaritmusát adja vissza valós számérték formájában.

(logand *int int ...*)

A számok listáján elvégzett bitenkénti logikai AND művelet eredményével tér vissza.

(logior *int int ...*)

A számok listáján elvégzett bitenkénti logikai inkluzív OR művelet eredményével tér vissza.

(lsh *int numbits*)

Egy szám megadott számú bittel történő, bitenkénti logikai eltolásával nyert értékét adja vissza.

(max *number number* ...)

Az adott számok közül a legnagyobbat adja vissza.

(min *number number* ...)

Az adott számok közül a legkisebbet adja vissza.

(minusp *number*)

Megvizsgálja, hogy a szám negatív-e.

(rem *num1 num2* ...)

Elosztja az első számot a másodikkal, és a maradékot adja vissza.

(sin *angle*)

Egy szög szinuszát adja eredményül radiánban kifejezett valós számként.

(sqrt *number*)

Egy szám négyzetgyökét adja vissza valós számként.

(zerop *number*)

Ellenőrzi, hogy egy szám értéke nulla-e.

Karakterlánc-kezelő függvények

A karakterlánc-kezelő függvények a következők:

(strcase *string [which]*)

A karakterlánc nagy- vagy kisbetűre konvertált változatát adja vissza.

(strcat *string1 [string2]* ...)

Egy karakterlánccal tér vissza, amely több karakterlánc összefűzéséből jött létre.

(strlen *[string]* ...)

Egy karakterláncban található karakterek számával tér vissza.

(substr *string start [length]*)

Egy karakterlánc részkarakterláncát adja vissza eredményül.

(wcmatch *string pattern*)

Globális karakter-összehasonlítást végez egy karakterláncon.

Egyenlőségi és feltételes függvények

Az egyenlőségi és feltételes függvények a következők:

(= *numstr [numstr]* ...)

A visszatérési érték T, ha az argumentumok értéke megegyezik; különben pedig nil.

(/= *numstr [numstr]* ...)

A visszatérési érték T, ha az argumentumok értéke nem egyezik meg, és nil, ha az argumentumok értéke megegyezik.

(< *numstr [numstr]* ...)

A visszatérési érték T, ha minden egyes argumentum értéke kisebb, mint a tőle jobbra álló argumentumé, különben pedig nil.

(<= *numstr [numstr]* ...)

A visszatérési érték T, ha minden egyes argumentum értéke kisebb vagy egyenlő, mint a tőle jobbra álló argumentumé, különben pedig nil.

(> *numstr [numstr]* ...)

A visszatérési érték T, ha minden egyes argumentum értéke nagyobb, mint a tőle jobbra álló argumentumé, különben pedig nil.

(>= *numstr [numstr]* ...)

A visszatérési érték T, ha minden egyes argumentum értéke nagyobb vagy egyenlő, mint a tőle jobbra álló argumentumé, különben pedig nil.

(and *expr* ...)

Egy kifejezéslista logikai AND értékével tér vissza.

(Boole *func int1 int2 ...*)

Általános bitenkénti Boolean függvényként szolgál.

(cond (*test1 result1 ...*) ...)

Az AutoLISP elsődleges feltételes függvényeként szolgál.

(eq *expr1 expr2*)

Két kifejezés azonosságát állapítja meg.

(equal *expr1 expr2 [fuzz]*)

Két kifejezés egyenlőségét állapítja meg.

(if *testexpr thenexpr [elseexpr]*)

Feltételesen értékeli ki a kifejezéseket.

(or *expr ...*)

Kifejezések listáján elvégzett logikai OR művelet eredményét adja vissza.

(repeat *int expr ...*)

Annyiszor értékel ki minden egyes kifejezést, amennyi a megadott szám értéke, és az utolsó kifejezés értékével tér vissza.

(while *testexpr expr ...*)

Kiértékeli a feltétel kifejezést, és ha annak értéke nem nil, akkor a többi kifejezést is kiértékeli; addig ismétli ezt a folyamatot, amíg a feltétel kifejezés értéke nil nem lesz.

Listakezelő függvények

A listakezelő függvények a következők:

(acad_strlsort *list*)

Ábécé-sorrendbe rendez egy karakterláncból álló listát.

(append *list ...*)

Tetszőleges számú listát fűz össze egy listává.

(assoc *item alist*)

Egy elemet keres egy asszociációs listában és az asszociációs lista bejegyzésével tér vissza.

(car *list*) és **(cdr *list*)**

Egy lista első elemével tér vissza, vagy egy olyan listával, amely az első elem kivételével tartalmazza az összes elemet.

(cons *new-first-element list*)

Legegyszerűbb listakészítő függvény.

(foreach *name list expr ...*)

Egy lista minden elemére kiértékeli a kifejezéseket.

(last *list*)

Egy lista utolsó elemével tér vissza.

(length *list*)

Egy listában szereplő elemek számát adja eredményül.

(list *expr ...*)

Listát hoz létre tetszőleges számú kifejezésből.

(listp *item*)

Megvizsgálja, hogy a tétel lista-e.

(mapcar *function list1 ... listn*)

Egy függvény végrehajtásakor kapott listával tér vissza, oly módon, hogy a lista vagy listák elemeit egyenként a függvény argumentumaiként kezeli.

(member *expr list*)

Egy kifejezés előfordulását keresi egy listában, és a listának a kifejezés első előfordulásával kezdődő maradékát adja vissza.

(nth *n list*)

Egy lista n-edik elemét adja vissza.

(reverse *list*)

Az elemeinek sorrendjét megfordítva ad vissza egy listát.

(subst *newitem olditem list*)

Másolatot ad vissza egy listáról, ahol a régi tétel minden előfordulása helyén az új tétel szerepel.

Szimbólumkezelő függvények

A szimbólumkezelő függvények a következők:

(atom *item*)

Ellenőrzi, hogy egy elem atom-e.

(atoms-family *format [symlist]*)

A pillanatnyilag definiált szimbólumok listájával tér vissza.

(boundp *sym*)

Ellenőrzi, hogy egy szimbólumnak van-e értéke.

(not *item*)

Megvizsgálja, hogy egy tétel nil értékű-e.

(null *item*)

Megvizsgálja, hogy a tételhez nil érték tartozik-e.

(numberp *item*)

Megvizsgálja, hogy a tétel egy valós vagy egész szám-e.

(quote *expr*)

A kifejezés kiértékeletlen formájával tér vissza.

(read *[string]*)

Egy karakterlánc első listáját vagy atomját adja vissza.

(set *sym expr*)

Egy aposztrofált szimbólum értékét egy kifejezéshez rendeli.

(setq *sym1 expr1 [sym2 expr2] ...*)

Egy vagy több szimbólum értékét társított kifejezésekhez rendeli.

(type *item*)

A megadott tétel típusát adja vissza.

Függvénykezelő függvények

A függvénykezelő függvények a következők:

(apply *function list*)

Egy argumentumlistát ad át egy megadott függvénynek.

(defun *sym argument-list expr ...*)

Egy függvényt definiál.

(eval *expr*)

Egy AutoLISP kifejezés kiértékelt eredményével tér vissza.

(lambda *arguments expr ...*)

Egy névtelen függvényt definiál.

(progn *[expr] ...*)

Egymás után kiértékeli mindegyik kifejezést, és az utolsó kifejezés értékével tér vissza.

(trace *function ...*)

Az AutoLISP hibakereső segédeszköze.

(untrace *function ...*)

Törli a megadott függvények nyomkövető flagjét.

Hibakezelő függvények

A hibakezelő függvények a következők:

(alert *string*)

Figyelmeztető ablakot jelenít meg karakterláncként megadott hiba- vagy figyelmeztető üzenettel.

(*error* *string*)

Felhasználó által definiálható hibakezelő függvény.

(exit)

Megszakítja az aktuális alkalmazást.

(quit)

Kilépteti az aktuális alkalmazást.

Alkalmazáskezelő függvények

Az alkalmazáskezelő függvények a következők:

(ads)

Az éppen betöltött ADS alkalmazások listájával tér vissza.

(arx)

Az éppen betöltött ADS alkalmazások listájával tér vissza.

(arxload *application [onfailure]*)

Betölt egy ARX alkalmazást.

(arxunload *application [onfailure]*)

Kitöröl egy ARX alkalmazást a memóriából.

(autoarxload *filename cmdlist*)

Parancsneveket definiál, amelyek egy asszociált ARX fájlt töltenek be.

(autoload *filename cmdlist*)

Parancsneveket definiál, amelyek egy asszociált AutoLISP fájlt töltenek be.

(autoxload *filename cmdlist*)

Parancsneveket definiál, amelyek egy asszociált ADS alkalmazást töltenek be.

(load *filename [onfailure]*)

Egy fájl AutoLISP kifejezéseit értékeli ki.

(startapp *appcmd file*)

Elindít egy Windows alkalmazást.

(xload *application [onfailure]*)

Betölt egy ADS alkalmazást.

(xunload *application [onfailure]*)

Töröl egy ADS alkalmazást a memóriából.

Segédfüggvények

A segédfüggvények lekérdező- és parancs, megjelenítésvezérlő, felhasználói bevitelre szolgáló, geometriai, konverziós, fájlkezelő és eszközelérési függvényekből állnak. Ezeket részletesebben a 8., "Általános segédfüggvények" című fejezet tárgyalja.

Lekérdező- és parancs függvények

A lekérdező- és parancs függvények a következők:

(acad_colordlg *colornum [flag]*)

Megjeleníti az AutoCAD színválasztó párbeszédpaneljét.

(acad_helpdlg *helpfile topic*)

Minden platformra érvényes Súgót hív (elavult).

(command *[arguments] ...*)

Egy AutoCAD parancsot hajt végre.

(getcfg *cfgname*)

Az acad.cfg fájl AppData szakaszának alkalmazás adataival tér vissza.

(getenv *variable-name*)

Egy környezeti változóhoz rendelt karakterlánc értékével tér vissza.

(getvar *varname*)

Egy AutoCAD rendszerváltozó értékét adja vissza.

(help *[helpfile [topic [command]]]*)

Meghívja a Súgót (az összes platformon).

(setcfg *cfgname cfgval*)

Alkalmazási adatokat ír az acad.cfg fájl AppData szakaszába.

(setfunhelp *function [helpfile [topic [command]]]*)

Egy felhasználó által definiált parancsot jegyez be a Súgóba, azért, hogy a megfelelő súgófájl és téma kerüljön meghívásra, amikor a felhasználó segítséget kér az adott paranccsal kapcsolatban.

(setvar *varname value*)

Egy AutoCAD rendszerváltozót megadott értékre állít.

(ver)

Az AutoLISP aktuális verziószámát tartalmazó karakterlánccal tér vissza.

Megjelenítésvezérlő függvények

A megjelenítésvezérlő függvények a következők:

(graphscr)

Az AutoCAD grafikus képernyőjét jeleníti meg.

(grclear)

Letörli az aktuális nézetablakot.

(grdraw *from to color [highlight]*)

Egy vektort rajzol két pont közé az aktuális nézetablakban.

(grtext *[box text [highlight]]*)

Egy szöveget ír az állapotsorba vagy a képernyő menüterületeire.

(grvecs *vlist [trans]*)

Vektorokat rajzol a grafikus képernyőre.

(menucmd *string*)

Menü parancsokat ad ki, vagy beállítja és visszaadja a menüpont státuszát.

(prin1 *[expr [file-desc]]*)

Egy kifejezést ír a parancssorba vagy egy nyitott fájlba.

(princ *[expr [file-desc]]*)

Egy kifejezést ír a parancssorba vagy egy nyitott fájlba.

(print *[expr [file-desc]]*)

Egy kifejezést ír a parancssorba vagy egy nyitott fájlba.

(prompt *msg*)

Egy karakterláncot jelenít meg a képernyő prompt területén.

(redraw *[ename [mode]]*)

Frissíti az aktuális nézetablakot vagy egy megadott objektumot (rajzelemet) az aktuális nézetablakban.

(terpri)

Egy újsor karaktert ír a parancssorba.

(textpage)

Letörli az AutoCAD szöveges képernyőjét, és megjeleníti azt az egyképernyős AutoCAD konfigurációk esetében.

(textscr)

Egyképernyős rendszerek esetében a grafikus képernyőről átvált a szöveges képernyőre (hasonlóan, mint az AutoCAD Képváltó funkcióbillentyűje).

(vports)

Az aktuális nézetablak-konfiguráció nézetablak-leíróinak listájával tér vissza.

Felhasználói bevitelre szolgáló függvények

A felhasználói bevitelre szolgáló függvények a következők:

(entsel *[prompt]*)

Felkínálja a felhasználónak, hogy egy pont megadásával válasszon egy objektumot (rajzelemet).

(getangle *[pt] [prompt]*)

Vár, amíg a felhasználó megad egy szöget, majd a szög radiánban kifejezett értékével tér vissza.

(getcorner *pt [prompt]*)

Vár, amíg a felhasználó megadja egy négyszög második sarkát.

(getdist *[pt] [prompt]*)

Vár, amíg a felhasználó megad egy távolságot.

(getfiled *title default ext flags*)

Az AutoCAD fájlkiválasztásra szolgáló párbeszédpaneljében egy fájlnév megadását kéri a felhasználótól, és a fájl nevét adja vissza.

(getint *[prompt]*)

Vár, amíg a felhasználó egy egész értéket ad meg, és ezzel az egésszel tér vissza.

(getkword *[prompt]*)

Vár, amíg a felhasználó választ egy kulcsszót, és azt a kulcsszót adja vissza.

(getorient *[pt] [prompt]*)

Vár, amíg a felhasználó megad egy szöget, és azt a szöget adja vissza radiánban.

(getpoint *[pt] [prompt]*)

Vár, amíg a felhasználó megad egy pontot, és azzal a ponttal tér vissza.

(getreal *[prompt]*)

Vár, amíg a felhasználó megad egy valós számot, és azt a valós számot adja vissza.

(getstring *[cr] [prompt]*)

Vár, amíg a felhasználó megad egy karakterláncot, és azt a karakterláncot adja vissza.

(initget *[bits] [string]*)

Kulcsszavakat hoz létre a következő felhasználói bevitelre szolgáló függvény számára.

(nentsel *[prompt]*)

Arra kéri a felhasználót, hogy egy pont megadásával válasszon ki egy objektumot (rajzelemet), és hozzáférést biztosít egy összetett objektum definíciós adataihoz.

(nentselp *[prompt] [pt]*)

Hasonló a nentsel függvényhez, azonban itt nincs szükség felhasználói adatbevitelre.

Geometriai függvények

A geometriai függvények a következők:

(angle *pt1 pt2*)

Két végpontja által meghatározott vonal radiánban megadott szögével tér vissza.

(distance *pt1 pt2*)

Két pont közti háromdimenziós távolsággal tér vissza.

(inters *pt1 pt2 pt3 pt4 [onseg]*)

Megtalálja két vonal metszéspontját.

(osnap *pt mode*)

Egy háromdimenziós ponttal, mint egy adott pontra alkalmazott Tárgyraszter mód eredményével tér vissza.

(polar *pt angle distance*)

Az FKR háromdimenziós pontját adja vissza egy ponttól megadott szöggel és távolsággal.

(textbox *elist*)

Leméri egy adott szövegobjektum kiterjedését, és a szöveg köré rajzolható keret átlós koordinátáival tér vissza.

Konverziós függvények

A konverziós függvények a következők:

(angtof *string [mode]*)

Egy karakterlánccal megadott szöget konvertál át radiánban kifejezett (lebegőpontos) valós számmá.

(angtos *angle [mode [precision]]*)

Egy radiánban megadott szögértéket konvertál át karakterlánccá.

(ascii *string*)

Egy karakterlánc első karakterének ASCII kódjával (egy egész szám) tér vissza.

(atof *string*)

Egy karakterlánc valós számmá konvertált értékével tér vissza.

(atoi *string*)

Egy karakterlánc egész számmá konvertált értékével tér vissza.

(chr *integer*)

Egy ASCII karakterkódot képviselő egész számot egykarakteres karakterlánccá konvertál.

(cvunit *value from to*)

Egyik mértékegységben mért értéket konvertál át egy másik mértékegységbe.

(distof *string [mode]*)

Egy (lebegőpontos) valós számot ábrázoló karakterláncot konvertál át valós értékké.

(itoa *int*)

Egy egész szám karakterlánccá konvertált értékével tér vissza.

(rtos *number [mode [precision]]*)

Egy számot karakterlánccá alakít.

(trans *pt from to [disp]*)

Egyik koordináta-rendszerből egy másikba transzformál egy pontot (vagy elmozdulást).

Fájlkezelő függvények

A fájlkezelő függvények a következők:

(close *file-desc*)

Lezár egy nyitott fájlt.

(findfile *filename*)

Megkeresi az adott fájlt az AutoCAD könyvtár útvonalban.

(open *filename mode*)

Megnyit egy fájlt az AutoLISP I/O függvényei általi hozzáférésre.

(read-char *[file-desc]*)

Egy billentyűzet bemeneti pufferéből vagy egy nyitott fájlból beolvasott karakter decimális ASCII kódját adja vissza.

(read-line *[file-desc]*)

Egy karakterláncot olvas be a billentyűzetről vagy egy nyitott fájlból.

(write-char *num [file-desc]*)

Egy karaktert ír a képernyőre vagy egy nyitott fájlba.

(write-line *string [file-desc]*)

Egy karakterláncot ír a képernyőre vagy egy nyitott fájlba.

Eszközelérési függvények

Az eszközelérési függvények a következők:

(grread *[track] [allkeys [curtype]]*)

Értékeket olvas be az AutoCAD bármelyik beviteli eszközéről.

(tablet *code [row1 row2 row3 direction]*)

Lekérdezi és beállítja a digitalizáló eszköz (tábla) kalibrációját.

Az ASE felület függvényei

Az ASE felület függvényei ASE adminisztratív, csatoláselérési, csatolás kiválasztás-kezelő, típus- és hibakezelő függvényekből állnak. Ezeket részletesebben a 8., "Általános segédfüggvények" című fejezet tárgyalja.

ASE adminisztratív függvények

Az ASE adminisztratív függvények a következők:

(ase_docurrent)

Megjeleníti az aktuális ASE beállításokat.

(ase_dolist *[obj_path [attr]]*)

Adatbázis-objektum elérési útvonalak listájával tér vissza.

(ase_dostatus *[obj_path]*)

Az adott objektum státuszával tér vissza.

(ase_lpcreate *table_path lp_name key*)

Új csatolás útvonalat hoz létre.

(ase_lperase *lp_name*)

A csatolás útvonalnév által megadott csatolás útvonalat törli.

(ase_lpisupdatable *lp_name*)

Ellenőrzi, hogy a csatolás útvonal egészében frissíthető-e.

(ase_lpkey *lp_name*)

Az adott csatolás táblához tartozó kulcsoszlop nevekkel tér vissza.

(ase_lplist *[obj_path]*)

Csatolás útvonalnevek listájával tér vissza.

(ase_lppath *lp_name*)

Adott csatolás útvonalhoz tartozó tábla útvonallal tér vissza.

(ase_lprename *lp_oldname lp_newname*)

Átnevezi a csatolás útvonalat.

(ase_version)

Megjeleníti az aktuális ASE verziót.

ASE csatoláselérési függvények

Az ASE csatoláselérési függvények a következők:

(ase_linkcreate *link_data*)

Új csatolást létesít.

(ase_linkget *link_data*)

A csatolás-azonosítóval meghatározott csatolás adattal tér vissza.

(ase_linkremove *link_id*)

Megszünteti a csatolás-azonosítóval megadott csatolást.

(ase_linkupdate *link_data*)

Frissíti a meglévő csatolást.

ASE csatolás kiválasztás-kezelő függvények

Az ASE csatolás kiválasztás-kezelő függvények a következők:

(ase_lsadd *link_sel link_id*)

Csatolás-azonosítót ad a csatolás kiválasztáshoz.

(ase_lscmp *link_sel1 link_sel2*)

Összehasonlítja a csatolás kiválasztás azonosítóit.

(ase_lscopy *link_sel*)

A csatolás kiválasztást egy újabba másolja.

(ase_lscreate *link_filter*)

Adott csatolásszűrő szerinti csatolás kiválasztást hoz létre.

(ase_lsdel *link_sel link_num*)

Törli a csatolás pozíciót a csatolás kiválasztásból.

(ase_lsentsel *link_sel*)

A csatolás kiválasztásban található csatolásokkal összefüggő objektumok halmazával (rajzelemekkel) tér vissza.

(ase_lserase *link_sel*)

Törli a csatolásokat és csatolás kiválasztást.

(ase_lsfree *link_sel*)

Felszabadítja a csatolás kiválasztást.

(ase_lsget *link_sel link_num*)

A csatolás kiválasztásban szereplő adott pozíciójú csatolás-azonosítóval tér vissza.

(ase_lsintersect *link_sel link_sel_intersect*)

A megadottakon kívül minden csatolás-azonosítót töröl egy csatolás kiválasztásból.

(ase_lsintersectfilter *link_filter*)

A megadott csatolásszűrővel nem összefüggő azonosítókat törli a csatolás kiválasztásból.

(ase_lsisupdatable *link_sel*)

Ellenőrzi, hogy a csatolás kiválasztás összes csatolása frissíthető-e.

(ase_lslpnames *link_sel*)

A csatolás kiválasztás összefüggő csatolás útvonal neveinek listájával tér vissza.

(ase_lsmemb *link_sel link_id*)

Ellenőrzi, hogy a csatolás-azonosító a csatolás kiválasztáshoz tartozik-e.

(ase_lsqty *link_sel*)

A csatolás kiválasztásban található csatolások számával tér vissza.

(ase_lssubtract *link_sel link_sel_subtract*)

Adott csatolás-azonosítókat töröl egy csatolás kiválasztásból.

(ase_lsunite *link_sel link_sel_union*)

Egy csatolás kiválasztást ad a másikhoz.

(ase_lsxnames *link_sel*)

A csatolás kiválasztás összefüggő xref vagy blokk neveinek listájával tér vissza.

ASE segédfüggvények

Az ASE segédfüggvények a következők:

(ase_docmp *obj_path1 obj_path2 [name_code]*)

Adatbázis-objektum elérési útvonalak részeit hasonlítja össze.

(ase_dopathcode *obj_path*)

Egy adott útvonal számára visszakeresi az adatbázis-objektum definíciós kódját.

(ase_dopathmake *env_name [cat_name [schema_name [table_name [lp_name]]]]*)

Adatbázis-objektum útvonalat készít atomi nevekből.

(ase_dopathname *obj_path name_code*)

Egy adott névvel tér vissza az adatbázis-objektum útvonalból.

ASE hibakezelő függvények

Az ASE hibakezelő függvényei a következők:

(ase_errcode *err_num*)

ASE hibakóddal tér vissza.

(ase_errdsc *err_num*)

Az ASE hibát megadó kóddal tér vissza.

(ase_errmsg *err_num*)

Az ASE hibát leíró karakerlánccal tér vissza.

(ase_errqty)

A hibaleírók számával tér vissza.

Kiválasztási halmaz, objektum és szimbólumtábla függvények

A kiválasztási halmaz, objektum és szimbólumtábla függvények kiválasztási halmazokat, objektumokat, bővített adatokat és szimbólumtáblákat kezelő függvényekből állnak. Ezeket részletesebben a 9., "Kiválasztási halmazokat, objektumokat és szimbólumtáblákat kezelő függvények" című fejezet tárgyalja.

Kiválasztási halmaz kezelő függvények

A kiválasztási halmazokat kezelő függvények a következők:

(ssadd *[ename [ss]]*)

Hozzáad egy objektumot (rajzelemet) egy kiválasztási halmazhoz, vagy új kiválasztási halmazt hoz létre.

(ssdel *ename ss*)

Kitöröl egy objektumot (rajzelemet) egy kiválasztási halmazból.

(ssget *[mode] [pt1 [pt2]] [pt-list] [filter-list]*)

Objektumok (rajzelemek) kiválasztására kéri a felhasználót, és egy kiválasztási halmazzal tér vissza.

(sslength *ss*)

Egy kiválasztási halmazban lévő objektumok (rajzelemek) számát adja meg.

(ssmemb *ename ss*)

Megvizsgálja, hogy egy objektum (rajzelem) szerepel-e egy kiválasztási halmazban.

(ssname *ss index*)

Egy kiválasztási halmaz megadott sorszámú (indexű) elemének objektum (rajzelem) nevét adja vissza.

Objektumkezelő függvények

Az objektumkezelő függvények a következők:

(entdel *ename*)

Objektumokat (rajzelemeket) töröl vagy korábban törölt objektumokat állít vissza.

(entget *ename [applist]*)

Egy objektum (rajzelem) definíciós adatait adja vissza.

(entlast)

Az utolsó nem törölt fő objektum (rajzelem) nevét adja vissza a rajzon.

(entmake *[elist]*)

Egy új objektumot (rajzelemet) hoz létre a rajzon.

(entmod *elist*)

Megváltoztatja egy objektum (rajzelem) definíciós adatait.

(entnext *[ename]*)

A rajzon található következő objektum (rajzelem) nevével tér vissza.

(entupd *ename*)

Egy objektum (rajzelem) ábráját frissíti fel a képernyőn.

(handent *handle*)

Egy objektum (rajzelem) nevét adja vissza annak sorszáma alapján.

Bővített adatokat kezelő függvények

A bővített adatokat kezelő függvények a következők:

(regapp *application*)

Bejegyez egy alkalmazásnevet az aktuális AutoCAD rajzba a bővített objektumadatok felhasználásának előkészítésére.

(xdroom *ename*)

A rajzelem számára elérhető, bővített adatokat (Xdata) tartalmazó hely méretével tér vissza.

(xdsize *list*)

Annak a helynek a méretével tér vissza (bájtokban), amelyet egy lista elfoglalna, ha egy rajzelemhez bővített adatként hozzá lenne csatolva.

Szimbólumtábla- és szótárkezelő függvények

A szimbólumtábla- és szótárkezelő függvények a következők:

(dictnext *ename symbol [rewind]*)

Megkeresi egy szótárban a következő tételt.

(dictsearch *ename symbol [setnext]*)

Egy tételt keres egy szótárban.

(namedobjdict)

Az aktuális rajz megnevezett objektum szótárának rajzelem nevét adja vissza. Ez a név a rajzban szereplő összes nem grafikus objektum kiindulópontjául szolgál.

(snvalid *sym_name*)

Ellenőrzi a szimbólumtábla-nevet az érvényes karakterek szempontjából.

(tblnext *table-name [rewind]*)

Megkeresi egy szimbólumtábla következő tételét.

(tblobjname *table-name symbol*)

Egy megadott szimbólumtábla-bejegyzés elemnevét adja vissza.

(tblsearch *table-name symbol [setnext]*)

Egy szimbólumot keres egy szimbólumtáblában.

Programozható párbeszédpanel függvények

A programozható párbeszédpanel függvények párbeszédpanel nyitó és záró, illetve zónákat és attribútumokat, listaablakokat és legördülő listákat, ábrazónákat és alkalmazás-specifikus adatokat kezelő függvényekből állnak. Ezeket részletesebben a 10., "Párbeszédpanelek kezelése" című fejezet tárgyalja.

Párbeszédpanel nyitó és záró függvények

A párbeszédpanel nyitó és záró függvények a következők:

(done_dialog *[status]*)

Lezárja a párbeszédpanelt.

(load_dialog *dclfile*)

Betölt egy DCL fájlt.

(new_dialog *dlgname dcl_id [action [screen-pt]]*)

Egy új párbeszédpanelt indít, megjeleníti azt, és egy alapértelmezés szerinti akciót rendelhet hozzá.

(start_dialog)

Megjelenít egy párbeszédpanelt, és megindítja a felhasználói bevitel fogadását.

(term_dialog)

Valamennyi aktuális párbeszédpanelt lezárja, mintha a felhasználó mindegyiket elvetette volna.

(unload_dialog *dcl_id*)

Töröl a memóriából egy DCL fájlt.

Zóna- és attribútumkezelő függvények

A zóna- és attribútumkezelő függvények a következők:

(action_tile *key action-expression*)

Egy akciót határoz meg, amely akkor kerül kiértékelésre, amikor a felhasználó kiválasztja az adott zónát a párbeszédpanelben.

(get_attr *key attribute*)

Visszakeresi egy párbeszédpanel-attribútum DCL fájlban lévő értékét.

(get_tile *key*)

Visszakeresi a zóna pillanatnyi értékét.

(mode_tile *key mode*)

Beállítja a párbeszédpanel-zóna állapotát.

(set_tile *key value*)

Egy párbeszédpanel-zóna értékét állítja be.

Listaablakokat és legördülő listákat kezelő függvények

A listaablakokat és legördülő listákat kezelő függvények a következők:

(add_list *string*)

Hozzáad egy karakterláncot az éppen aktív párbeszédpanel listájához vagy módosítja azt.

(end_list)

Befejezi az éppen aktív párbeszédpanel-listával kapcsolatos műveleteket.

(start_list *key [operation [index]]*)

Megindítja a listaablak vagy legördülő lista párbeszédpanel zónájában található lista feldolgozását.

Ábrazóna-kezelő fügvények

Az ábrazóna-kezelő függvények a következők:

(dimx_tile *key*) and (dimy_tile *key*)

Egy zóna méreteit adja vissza párbeszédpanel-egységekben.

(end_image)

Befejezi az éppen aktív párbeszédpanel-ábra szerkesztését.

(fill_image *x1 y1 wid hgt color*)

Egy kitöltött téglalapot rajzol az éppen aktív párbeszédpanel ábrazónájába.

(slide_image *x1 y1 wid hgt sldname*)

Egy AutoCAD diát rajzol az éppen aktív párbeszédpanel ábrazónájába.

(start_image *key*)

Megindítja egy ábra szerkesztését a párbeszédpanel zónájában.

(vector_image *x1 y1 x2 y2 color*)

Egy vektort rajzol az éppen aktív párbeszédpanel-ábrába.

Alkalmazás-specifikus adatokat kezelő függvények

Az alkalmazás-specifikus adatokat kezelő függvények a következők:

(client_data_tile *key clientdata*)

Alkalmazás által kezelt adatokat társít egy párbeszédpanel-zónához.

Memóriakezelő függvények

A következő memóriakezelő függvényeket a 15., "Memóriakezelés" című fejezet részletesebben tárgyalja.

(alloc *int*)

Megadott számú csomópontra állítja be a szegmens méretét.

(expand *number*)

Egy adott számú szegmens által kért csomóponti (node) helyet jelöl ki.

(gc)

Hulladékgyűjtő folyamatot indít, amely felszabadítja a nem használt csomópontokat.

(mem)

Kijelzi az AutoLISP memóriájának aktuális állapotát.

(vmon)

Virtuális függvénylapozást tesz lehetővé (elavult).

13

AUTOLISP FÜGGVÉNYEK KATALÓGUSA

Az alábbiakban ábécé-sorrendben megtalálható az AutoCAD által definiált összes szabvány AutoLISP függvény.

Ebben a fejezetben rövid leírás található az összes függvény használatáról; a szintaktikai utasítások pedig jelzik az egyes függvényekhez járuló argumentumok sorrendjét és típusát. (További információk a szintaxisról az "AutoLISP függvények szintaxisa" című résznél, a 194. oldalon találhatók.) Ha nem ismeri a keresett függvény nevét, használja az "Az AutoLISP függvények áttekintése" című, 12. fejezetet, ahol funkcionális csoportokra osztva találja meg a függvényeket.

A *szám* típusú argumentumhoz kiegészítő információ szükséges: egy szám lehet valós vagy egész szám; illetve szimbólum, amelyhez valós vagy egész értéket rendeltünk. Ha az összes argumentum egész, akkor az eredmény is egész szám lesz. Ha az argumentumok bármelyike valós szám, az egészek is valós számmá konvertálódnak, és az eredmény is valós szám lesz.

A legtöbb AutoLISP függvény mindig elérhető. Néhányat azonban egy ADS vagy ARX alkalmazás definiál. A külsőleg definiált függvényeket tartalmazó fájl a "Külsőleg definiált függvény" cím alatt található. Mielőtt használnánk ezeket a függvényeket ellenőrizhetjük, hogy elérhetőek-e. További információ az "AutoLISP alkalmazások használata" című résznél, a 161. oldalon található.

+ (összeadás)

A számok összegével tér vissza

(+ *[number number]* ...)

Ha csak egy *number* argumentumot ad meg, a függvény ezt a nullához adja hozzá; magát a számot adja vissza. Argumentum nélkül nullát ad vissza.

```
(+ 1 2)                 eredménye 3
(+ 1 2 3 4.5)           eredménye 10.5
(+ 1 2 3 4.0)           eredménye 10.0
```

– (kivonás)

Az első számból kivonja a következő számok összegét, és a különbséggel tér vissza

(- *[number number]* ...)

Ha két *number* argumentumnál többet ad meg, a függvény kivonja az első számból az utána következő számok összegét. Ha csak egy *number* argumentumot ad, a függvény azt nullából vonja ki; a számot adja vissza. Argumentum nélkül nullát ad vissza.

```
(- 50 40)               eredménye 10
(- 50 40.0)             eredménye 10.0
(- 50 40.0 2.5)         eredménye 7.5
(- 8)                   eredménye -8
```

* (szorzás)

A számok szorzatával tér vissza

(* *[number number]* ...)

Ha csak egy *number* argumentumot adunk meg, a függvény azt eggyel szorozza meg; a számot adja vissza. Argumentum nélkül nullát ad vissza.

```
(* 2 3)                 eredménye 6
(* 2 3.0)               eredménye 6.0
(* 2 3 4.0)             eredménye 24.0
(* 3 -4.5)              eredménye -13.5
(* 3)                   eredménye 3
```

/ (osztás)

Az első számot elosztja a többi szám szorzatával, és a hányadossal tér vissza

```
(/ [number number] ...)
```

Ha két *number* argumentumnál többet ad meg, a függvény elosztja az első számot a többi szám szorzatával, és a hányadossal tér vissza. Egyetlen *number* argumentum megadása esetén a függvény azt eggyel osztja el; magát a számot adja vissza. Argumentum nélkül nullát ad vissza.

```
(/ 100 2)              eredménye 50
(/ 100 2.0)            eredménye 50.0
(/ 100 20.0 2)         eredménye 2.5
(/ 100 20 2)           eredménye 2
(/ 4)                  eredménye 4
```

= (egyenlő)

A visszatérési érték `T`, ha az argumentumok értéke megegyezik; különben pedig `nil`

```
(= numstr [numstr] ...)
```

Minden *numstr* argumentum lehet szám vagy karakterlánc.

```
(= 4 4.0)              eredménye T
(= 20 388)             eredménye nil
(= 2.4 2.4 2.4)        eredménye T
(= 499 499 500)        eredménye nil
(= "én" "én")          eredménye T
(= "én" "te")          eredménye nil
```

További tudnivalók

eq és **equal** függvény

/= (nem egyenlő)

A visszatérési érték T, ha az argumentumok értéke nem egyezik meg, és nil, ha az argumentumok értéke megegyezik

(/= *numstr [numstr]* ...)

Minden *numstr* argumentum lehet szám vagy karakterlánc.

```
(/= 10 20)              eredménye T
(/= "te" "te")          eredménye nil
(/= 5.43 5.44)          eredménye T
(/= 10 20)              eredménye T
```

< (kisebb, mint)

A visszatérési érték T, ha minden egyes argumentum értéke kisebb, mint a tőle jobbra álló argumentumé, különben pedig nil

(< *numstr [numstr]* ...)

Minden *numstr* argumentum lehet szám vagy karakterlánc.

```
(< 10 20)               eredménye T
(< "b" "c")             eredménye T
(< 357 33.2)            eredménye nil
(< 2 3 88)              eredménye T
(< 2 3 4 4)             eredménye nil
```

<= (kisebb vagy egyenlő)

A visszatérési érték T, ha minden egyes argumentum értéke kisebb vagy egyenlő, mint a tőle jobbra álló argumentumé, különben pedig nil

(<= *numstr [numstr]* ...)

Minden *numstr* argumentum lehet szám vagy karakterlánc.

```
(<= 10 20)              eredménye T
(<= "b" "b")            eredménye T
(<= 357 33.2)           eredménye nil
(<= 2 9 9)              eredménye T
(<= 2 9 4 5)            eredménye nil
```

> (nagyobb, mint)

A visszatérési érték `T`, ha minden egyes argumentum értéke nagyobb, mint a tőle jobbra álló argumentumé, különben pedig `nil`

(> *numstr [numstr] ...*)

Minden *numstr* argumentum lehet szám vagy karakterlánc.

```
(> 120 17)          eredménye T
(> "c" "b")         eredménye T
(> 3.5 1792)        eredménye nil
(> 77 4 2)          eredménye T
(> 77 4 4)          eredménye nil
```

>= (nagyobb vagy egyenlő)

A visszatérési érték `T`, ha minden egyes argumentum értéke nagyobb vagy egyenlő, mint a tőle jobbra álló argumentumé, különben pedig `nil`

(>= *numstr [numstr] ...*)

Minden *numstr* argumentum lehet szám vagy karakterlánc.

```
(>= 120 17)         eredménye T
(>= "c" "c")        eredménye T
(>= 3.5 1792)       eredménye nil
(>= 77 4 4)         eredménye T
(>= 77 4 9)         eredménye nil
```

~ (bitenkénti NOT)

Az argumentumon bitenként elvégzett NOT (egyes komplemens) művelet értékével tér vissza

(~ *egész*)

```
(~ 3)               eredménye -4
(~ 100)             eredménye -101
(~ -4)              eredménye  3
```

1+ (növelés)

Az argumentum eggyel megnövelt értékével tér vissza

(1+ *number*)

```
(1+ 5)                  eredménye   6
(1+ -17.5)              eredménye   -16.5
```

1– (csökkentés)

Az argumentum eggyel csökkentett értékével tér vissza

(1- *number*)

```
(1- 5)                  eredménye   4
(1- -17.5)              eredménye   -18.5
```

abs

Az argumentum abszolút értékével tér vissza

(abs *number*)

```
(abs 100)               eredménye   100
(abs -100)              eredménye   100
(abs -99.25)            eredménye   99.25
```

acad_colordlg

Megjeleníti az AutoCAD színválasztó párbeszédpaneljét

(acad_colordlg *colornum [flag]*)

A *colornum* argumentum egy 0 és 255 közötti egész szám, amely meghatározza, hogy az AutoCAD melyik színt jelenítse meg kezdeti alapértelmezésként. A *colornum* 0 értéke esetén a BLOKK, 256 esetén pedig a FÓLIA lesz alapértelmezett. Az opcionális *flag* argumentum nil értékre állítása letiltja a FÓLIA és BLOKK nyomógombokat. A *flag* argumentum elhagyása vagy nil-től eltérő értéke engedélyezi a FÓLIA és BLOKK nyomógombokat.

Az **acad_colordlg** függvény a felhasználó által választott színkóddal tér vissza. Ha a felhasználó elveti a párbeszédpanelt, akkor az **acad_colordlg** függvény nil értékkel tér vissza.

Az alábbi függvényhívás felajánlja a felhasználónak, hogy válasszon egy színt, és a zöldet adja meg alapértelmezésként:

```
(acad_colordlg 3)
```

Külsőleg definiált függvény

acadapp ADS alkalmazás

acad_helpdlg

Minden platformra érvényes Súgót hív (elavult)

(acad_helpdlg *helpfile topic*)

Ezt a külsőleg definiált függvényt felváltotta a beépített **help** függvény. Az AutoCAD korábbi verzióival való kompatibilitás miatt jelent meg itt is. Részletes leírása a **"help"** függvénynél található .

Külsőleg definiált függvény

acadr13.lsp AutoLISP által definiált függvény

acad_strlsort

Ábécé-sorrendbe rendez egy karakterláncból álló listát

(acad _strlsort *list*)

A *list* argumentum nem más, mint a rendezni kívánt karakterlánc-lista. Az **acad_strlsort** függvény ezt a listát adja vissza ábécé-sorrendbe rendezve. Ha a *list* argumentumlista nem érvényes, vagy nincs elég memória a rendezésre, az **acad_strlsort** függvény **nil** értékkel tér vissza.

A következő program a napok neveiből álló listát rendezi:

```
(setq mos '("hétfő" "kedd" "szerda" "csütörtök"
            "péntek" "szombat" "vasárnap" ))
(acad_strlsort mos)
```

Az alábbi listával tér vissza:

```
("csütörtök" "hétfő" "kedd" "péntek"
 "szerda" "szombat" "vasárnap" )
```

Külsőleg definiált függvény

acadapp ADS alkalmazás

action_tile

Egy akciót határoz meg, amely akkor kerül kiértékelésre, amikor a felhasználó kiválasztja az adott zónát a párbeszédpanelben

(action_tile *key action-expression*)

A *key* és az *action-expression* argumentumok karakterláncok. A *key* argumentum az akciót kiváltó zóna neve (a zóna `key` attribútumában megadott név). A *key* argumentum különbséget tesz a kis- és nagybetűk között. Az *action-expression* (akciókifejezés) akkor kerül kiértékelésre, amikor a zónát a felhasználó kiválasztja.

Az **action_tile** által hozzárendelt akció felülírja a párbeszédpanel alapértelmezés szerinti (a **new_dialog** által hozzárendelt) akcióját vagy a zóna `action` attribútumát, amennyiben ezek adottak. A kifejezés hivatkozhat a zóna aktuális értékére (`value` attribútumára) mint `$value`, nevére mint `$key`, alkalmazás-specifikus adatára (amelyet a **client_data_tile** állít be) mint `$data`, visszajelzési indítékára mint `$reason`, és -- amennyiben a zóna egy ábra-nyomógomb -- a koordinátáira mint `$x` és `$y`.

További tudnivalók

"Alapértelmezés szerinti és DCL akciók" című résznél, a 309. oldalon

add_list

Hozzáad egy karakterláncot az éppen aktív párbeszédpanel listájához vagy módosítja azt

(add_list *string*)

Az **add_list** függvény használata előtt a **start_list** függvénnyel meg kell nyitni és inicializálni kell a listát. A **start_list** függvényben megadott művelettől függően a *string* argumentum vagy hozzáadódik az aktuális listához, vagy felváltja a lista aktuális elemét.

Feltéve, hogy az éppen aktív DCL fájl rendelkezik egy `hosszúlista` kulcsú legördülő listával vagy listaablakkal, a következő kódrészlet inicializálja a listát és hozzáadja az `hlista` nevű listában levő szövegeket.

```
(setq hlista '("első sor" "második sor" "harmadik sor"))
(start_list "hosszúlista")
(mapcar 'add_list hlista)
(end_list)
```

Miután definiáltuk a listát, a következő programrészlet kicseréli a második sor szövegét "2. sor"-ra.

```
(start_list "hosszúlista" 1 0)
(add_list "2.sor")
(end_list)
```

További tudnivalók

start_list és **end_list** függvény

ads

Az éppen betöltött ADS alkalmazások listájával tér vissza

(ads)

Minden alkalmazás és elérési útvonala idézőjelben szerepel.

(ads) *eredménye lehet* ("fajl/prog/PROG1" "PROG2")

További tudnivalók

xload és **xunload** függvény

alert

Figyelmeztető ablakot jelenít meg karakterláncként megadott hiba- vagy figyelmeztető üzenettel

(alert *string*)

A figyelmeztető ablak egy párbeszédpanel, egyetlen OK gombbal.

```
(alert "Ez a függvény nem elérhető.")
```

Több sort is megjeleníthet, ha az újsor karaktert használja a *string* argumentumban.

```
(alert "Ez a függvény\nnem elérhető.")
```

Megjegyzés A sor hossza és a sorok száma egy figyelmeztető ablakban függ a platformtól, eszköztől, és az ablaktól. Az AutoCAD csonkolja azt a karakterláncot, amely nem fér el egy figyelmeztető ablakban.

alloc

Megadott számú csomópontra állítja be a szegmens méretét

(alloc *int*)

Ez a függvény a korábbi szegmensmérettel tér vissza.

További tudnivalók

"Manuális allokáció" című résznél, a 566. oldalon

and

Egy kifejezéslista logikai AND értékével tér vissza

(and *expr* ...)

Ha bármelyik kifejezés értéke `nil`, a függvény abbahagyja a további értékelést és `nil` értékkel tér vissza, különben a visszatérési értéke `T`.

Például a

```
(setq a 103 b nil c "karakterlánc")
```

értékadás után

```
(and 1.4 a c)          eredménye  T
(and 1.4 a b c)        eredménye  nil
```

angle

Két végpontja által meghatározott vonal radiánban megadott szögével tér vissza

(angle *pt1 pt2*)

A függvény az aktuális szerkesztősík *X* tengelyétől méri a szöget radiánban. A szög értéke az óra járásával ellentétes irányban növekszik. Ha háromdimenziós pontokat adunk meg, a függvény az aktuális szerkesztősíkra eső vetületükkel számol.

```
(angle '(1.0 1.0) '(1.0 4.0))     eredménye  1.5708
(angle '(5.0 1.33) '(2.4 1.33)    eredménye  3.14159
```

További tudnivalók

"Szögkonverziók" című résznél, a 246. oldalon

angtof

Egy karakterlánccal megadott szöget konvertál át radiánban kifejezett (lebegőpontos) valós számmá

(angtof *string [mode]*)

A `string` argumentum egy szöget ír le a `mode` argumentum által megadott forma szerint. Az alábbi táblázatban bemutatott `mode` argumentum megadja, hogy a karakterlánc milyen mértékegységben adja meg a szöget. Az értéknek meg kell felelnie az AutoCAD AUNITS rendszerváltozója által megengedett értékeknek. Ha a `mode` argumentumot elhagyjuk, a függvény az AUNITS rendszerváltozó aktuális értékét használja.

Szög mértékegységek értékei

Mode értéke	*A string formátuma*
0	Fok
1	Fok/perc/másodperc
2	Grad
3	Radián
4	Geodéziai mértékegységek

A `string` argumentumnak olyan karakterláncnak kell lennie, amelyet az **angtof** függvény helyesen tud elemezni a `mode` argumentum által megadott formában. Ez lehet az **angtos** függvény által visszaadott, vagy az AutoCAD által megengedett, billentyűzetről bevitt karakterlánc forma.

Az **angtof** és **angtos** függvények kiegészítik egymást: ha az **angtof** függvénynek egy **angtos** függvény által visszaadott karakterláncot ad meg, az **angtof** garantáltan érvényes értékkel tér vissza, és fordítva (feltéve, hogy a `mode` értékei megegyeznek).

Ha az **angtof** függvény végrehajtása eredményes volt, egy radián értékkel tér vissza; különben a visszatérési érték `nil`.

angtos

Egy radiánban megadott szögértéket konvertál át karakterlánccá

(angtos *angle [mode [precision]]*)

Az **angtos** függvény az *angle* argumentumot, amely egy valós számmal radiánban megadott szög, karakterlánccá alakítja a *mode*, a *precision* argumentumoknak, valamint az AutoCAD UNITMODE rendszerváltozójának és a DIMZIN méretváltozó beállításainak megfelelően. A *mode* és *precision* argumentumok egész számok, amelyek a szög mértékegységének fajtáját és a pontosságot adják meg.

Szög mértékegység értékek

Mode értéke	*A karakterlánc formátuma*
0	Fok
1	Fok/perc/másodperc
2	Grad
3	Radián
4	Geodéziai mértékegységek

A *precision* olyan egész típusú argumentum, amely a kívánt pontossággal megadja a tizedesjegyek számát. A *mode* és *precision* argumentumok az AutoCAD AUNITS és AUPREC rendszerváltozóinak felelnek meg. Ha elhagyja ezeket az argumentumokat, az **angtos** az AUNITS és AUPREC beállításait használja .

Az **angtos** függvény elfogad negatív *angle* argumentumot is, de mindig átalakítja azt egy nulla és 2 pi közötti pozitív számmá, mielőtt elvégzi a megadott konverziót.

```
(angtos 0.785398 0 4)      eredménye "45.0000"
(angtos -0.785398 0 4)     eredménye "315.0000"
```

A UNITMODE változónak akkor van hatása a visszaadott karakterláncra, ha a Ön a geodéziai mértékegységet választja (a *mode* értéke 4). Ha a UNITMODE = 0, akkor szóközöket tesz a karakterláncba (például "É 45f K"); ha UNITMODE = 1, akkor nincs szóköz a karakterláncban (például "É45dK").

Megjegyzés Azoknak a rutinoknak, amelyek tetszőleges (tehát nem az ANGBASE értékéhez viszonyított) szög megjelenítésére használják az **angtos** függvényt, az ANGBASE rendszerváltozó értékét kell ellenőrizniük és figyelembe venniük .

További tudnivalók

"Karakterlánc-konverziók" című résznél, a 243. oldalon

append

Tetszőleges számú listát fűz össze egy listává

(append *list* ...)

```
(append '(a b) '(c d))          eredménye (A B C D)
(append '((a)(b)) '((c)(d)))    eredménye ((A)(B)(C)(D))
```

apply

Egy argumentumlistát ad át egy megadott függvénynek

(apply *function list*)

Az **apply** függvény egyaránt működik a beépített (subrs) és felhasználó által definiált (a **defun** vagy a **lambda** által készített) függvénnyekkel is .

```
(apply '+ '(1 2 3))                 eredménye 6
(apply 'strcat '("a" "b" "c"))      eredménye "abc"
```

arx

Az éppen betöltött ARX alkalmazások listájával tér vissza

(arx)

Minden egyes alkalmazás és annak elérési útvonala idézőjelek között szerepel.

```
(arx)   lehetséges eredménye   ("fajl/prog/PROG1" "PROG2")
```

További tudnivalók

arxload és **arxunload** függvény

arxload

Betölt egy ARX alkalmazást

(arxload *application [onfailure]*)

Az *application* argumentumot idézőjelbe tett karakterlánccal vagy egy olyan változóval kell megadni, amely tartalmazza a futtatható fájl nevét. A fájl betöltésekor a rendszer ellenőrzi, hogy érvényes ARX alkalmazásról van-e szó.

Az **arxload** művelet sikertelensége rendszerint AutoLISP hibát okoz. Azonban, ha az *onfailure* argumentumot is megadjuk, akkor az **arxload** hiba esetén ennek az értékével tér vissza hibaüzenet helyett.

Az alkalmazás sikeres betöltése esetén a függvény az alkalmazás nevével tér vissza .

```
(arxload "/sajatalk/alkx")
```
ha sikeres, eredménye

"/sajatalk/alkx"

Ha egy olyan alkalmazást kíván betölteni, amely már a memóriában van, az **arxload** a következő üzenetet adja, és az alkalmazás nevével tér vissza.

Az "*alkalmazás*" alkalmazás már be van töltve.

Az **arxload** használata előtt az **arx** függvénnyel ellenőrizheti az éppen betöltött alkalmazásokat .

arxunload

Kitöröl egy ARX alkalmazást a memóriából

(arxunload *application [onfailure]*)

Az alkalmazás memóriából való sikeres törlése esetén az alkalmazás nevével tér vissza, különben pedig hibaüzenetet ad.

Az *application* argumentumot idézőjelbe tett karakterlánccal vagy egy olyan változóval adja meg, amely tartalmazza az **arxload** által betöltött alkalmazás nevét. Az alkalmazás nevét pontosan úgy kell beírni, mint azt az **arxload** függvénynél tette. Ha az alkalmazás elérési útvonalát (könyvtárnevet) vagy kiterjesztését az **arxload** függvénynél már megadta, azt az **arxunload** függvénynél elhagyhatja.

Például a következő függvényhívás törli a memóriából az alkx alkalmazást, melyet korábban az **arxload** függvénnyel töltöttünk be:

```
(arxunload "alkx")
```
ha sikeres, eredménye `"alkx"`

Az **arunxload** művelet sikertelensége rendszerint AutoLISP hibát okoz. Azonban, ha az *onfailure* argumentumot is megadja, akkor az **arunxload** hiba esetén ennek az értékével tér vissza hibaüzenet helyett. Ez a jellegzetesség megegyezik az **arxload** függvénynél tárgyaltakkal.

ase_docmp

Adatbázis-objektum elérési útvonalak részeit hasonlítja össze

(ase_docmp *obj_path1 obj_path2 [name_code]*)

Az *obj_path1* és *obj_path2* argumentumok karakterlánc típusúak, amelyek az összehasonlítandó útvonalakat adják meg. A *name_code* argumentum egy egész szám, amely megadja, hogy az útvonal melyik részét hasonlítsa össze a függvény. Az **ase_dopathname** függvénynél megtalálható az érvényes értékeket tartalmazó táblázat. Ha a *name_code* argumentumot elhagyjuk, a függvény a teljes útvonalat összehasonlítja (a kód értéke 6). A kiválasztott részek egyezése esetén T értékkel tér vissza, különben pedig a visszatérési érték nil.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_docurrent

Megjeleníti az aktuális ASE beállításokat

(ase_docurrent)

Sikeres működés esetén egy karakterlánccal tér vissza, amely az aktuális ASE beállításokat mutatja. Hiba esetén, vagy ha egyetlen adatbázis-objektum referencia sincs beállítva, a visszatérési érték nil .

Külsőleg definiált függvény

ase ARX alkalmazás

ase_dolist

Adatbázis-objektum elérési útvonalak listájával tér vissza

(ase_dolist *[obj_path [attr]]*)

Az `obj_path` argumentum egy karakterlánc, amely megadja az adatbázis-objektum hierarchiaszintjét. A függvény az alárendelt szintű adatbázis-objektumnevek listájával tér visza. Ha a környezeten belüli katalógusoknak akar útvonal listát küldeni, akkor meg kell adni egy karakterláncot, amely a visszaadott objektum attribútum értékeit meghatározó környezetet (`name.attr`) tartalmazza. Ha az `objpath` argumentum egy üres karakterlánc, vagy ha nincs megadva, a környezetek litájával tér vissza a függvény. Az `attr` argumentum egy egész szám, amely a visszaadott objektumok attribútum értékeit adja meg.

attr bit értékek

Érték	*Leírás*
1	Az adatbázis-objektum aktuális
2	Az adatbázis-objektum bejegyzett
4	Az adatbázis-objektum létezik az SQL adatbázisban
8	Az adatbázis-objektum a kapcsolt környezethez tartozik

Az **ase_dolist** függvény visszadja az összes elérhető, bejegyzett és nem elérhető adatbázis-objektum listáját. Az összes elérhető útvonalat visszaadja, ha az `attr` argumentumot nem adjuk meg. Sikeres működés esetén egy karakterlánc listával tér vissza. Hiba esetén, vagy ha nincs az `attr` argumentum maszkjával megegyező alárendelt adatbázis-objektum (például, ha a táblareferencia meg van adva) a visszatérési érték `nil`.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_dopathcode

Egy adott útvonal számára visszakeresi az adatbázis-objektum definíciós kódját

(ase_dopathcode *obj_path*)

Az *obj_path* argumentum egy karakterlánc, amely az adatbázis-objektum hierarchiaszintjét adja meg. Ez a függvény sikeres működés esetén az alábbi táblázatnak megfelelő egész számot adja vissza, különben pedig a visszatérési érték `nil` .

Adatbázis-objektum definíciós kódok

Kód	*Útvonal elem*
0	Ismeretlen
1	Környezet
2	Katalógus
3	Séma
4	Tábla
5	Csatolás útvonalnév
6	Teljes útvonal (csatolás útvonalnévvel)
7	Adatbázis útvonal (csatolás útvonalnév nélkül)
8	SQL tábla útvonal

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_dopathmake

Adatbázis-objektum útvonalat készít atomi nevekből

```
(ase_dopathmake env_name [cat_name [schema_name
   [table_name [lp_name]]]])
```

Az összes argumentum karakterlánc típusú. Sikeres hívás esetén egy teljes adatbázis-objektum útvonallal, különben `nil` értékkel tér vissza.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_dopathname

Egy adott névvel tér vissza az adatbázis-objektum útvonalból

```
(ase_dopathname obj_path name_code)
```

Az `obj_path` egy karakterlánc, amely az adatbázis-objektum hierarchiaszintjét adja meg . A `name_code` egy egész szám, amely az alábbi táblázattal összhangban a visszatérő név jellegét adja meg.

Adatbázis-objektum definíciós kódok

Kód	*Útvonal elem*
0	Ismeretlen
1	Környezet
2	Katalógus
3	Séma
4	Tábla
5	Csatolás útvonalnév
6	Teljes útvonal (csatolás útvonalnévvel)
7	Adatbázis-objektum útvonal (csatolás útvonalnév nélkül)

Adatbázis-objektum definíciós kódok (*folytatás*)

Kód	*Útvonal elem*
8	SQL tábla útvonal

Sikeres működés esetén az adott név, különben `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_dostatus

Az adott objektum státuszával tér vissza

`(ase_dostatus [obj_path])`

Az `obj_path` az adott adatbázis-objektum útvonalát megadó karakterlánc. Ha az `obj_path` nincs megadva, vagy egy üres karakterlánc, akkor ezzel a függvénnyel az aktuális beállítások, illetve bármelyik csatolás útvonal létezését lehet ellenőrizni. Sikeres működés esetén egy egész számot ad vissza, különben a visszatérési érték `nil`. A visszaadott érték az alábbi táblázatban található számok összege.

obl_path visszatérési értékek

Érték	*Leírás*
1	Az objektum aktuális
2	Az objektum bejegyzett
4	Az objektum hozzáférhető (létezik az ASI környezetben)
8	Az objektum a csatolt környezethez tartozik (az ASE felhasználói felület segítségével)
16	Nem lehet megállapítani, hogy az objektum létezik-e az adatbázisban
32	Néhány kulcsoszlop nem létezik az adatbázisban (csak csatolás útvonalnevekkel rendelkező táblák esetén)
64	A kulcsoszlop adat definíciója a rajzon eltér az igazi táblában levőtől (csak csatolás útvonalnevekkel rendelkező táblák esetén)

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_errcode

ASE hibakóddal tér vissza

(ase_errcode *err_num*)

Az *err_num* argumentum egy egész szám, amely megadja a hibaleíró sorszámát. Ez a függvény egy hibakóddal tér vissza. Az ASE hibakódok az *AutoCAD Devolopers Guide* "Syntax of SQL2 Implementation" című C mellékletében találhatók.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_errdsc

Az ASE hibát megadó kóddal tér vissza

(ase_errdsc *err_num*)

Az *err_num* argumentum egy egész szám, amely megadja a hibaleíró sorszámát. Ez a függvény egy egész számmal tér vissza, amely jelzi a hiba okát. Az ASE hibakódok listája az *AutoCAD Devolopers Guide* "Syntax of SQL2 Implementation" című C mellékletében található.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_errmsg

Az ASE hibát leíró karakerlánccal tér vissza

(ase_errmsg *err_num*)

Az *err_num* argumentum egy egész szám, amely megadja a hibaleíró sorszámát. Ez a függvény a hibát leíró karakterlánccal tér vissza. Az **ase_errmsg** függvény visszatérési értéke `nil`, ha az *err_num* nem

érvényes. Az ASE hibakódok listája az *AutoCAD Devolopers Guide* "Syntax of SQL2 Implementation" című C mellékletében található.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_errqty

A hibaleírók számával tér vissza

(ase_errqty)

Ha nincs hiba, az **ase_errqty** függvény visszatérési értéke 0.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_linkcreate

Új csatolást létesít

(ase_linkcreate *link_data*)

A *link_data* argumentum egy karakterlánc, melynek formátuma a következő:

csatolás_attr1.attr_érték1 csatolás_attr2.attr_érték2 ...

A csatolás attribútumok és attribútumértékek párjaiból áll. A *link_attr* egy egész szám, amely megadja az attribútum típusát. A következő táblázatban találhatók az érvényes egész szám értékek.

link_attr értékek

Érték	*Leírás*
1	Csatolás-azonosító (egy szerkesztési alkalomra érvényes)
2	Csatolás típus kód
3	Csatolás státusz
4	Rajzelemnév
5	Csatolás útvonalnév
6	Kulcsérték

link_attr értékek (*folytatás*)

Érték	*Leírás*
7	Megjeleníthető attribútum oszlop nevek (a megjeleníthető attribútum csatolás számára)
8	Megjeleníthető rajzelem paraméterek (csak a megjeleníthető attribútum csatolás létrehozására)
9	Megjeleníthető attribútum oszlop értékek (csak a megjeleníthető attribútum csatolás létrehozására)
10	Külső referencia (Xref) vagy blokknév
11	Foglalt belső attribútum (a csatolás létrehozásakor és változtatásakor figyelmen kívül hagyja)

Az ASE felület a csatolások két típusát támogatja: a rajzelem csatolásokat és a megjeleníthető attribútumokat. A csatolás típusát a csatolás létrehozásakor meg kell adni.

A rajzelem csatolás kódja 1. A megjeleníthető attribútum csatolások kódja 2. Minden egyes csatolás rendelkezik egy státusszal, amely megadja, hogy a csatolás frissíthető-e. Amennyiben a csatolás attribútum 0 vagy páratlan, akkor a csatolás nem frissíthető.

Minden meglévő csatolásnak van azonosítója, amelyet a csatolás rajzból való betöltésekor vagy a csatolás létrehozása után rendelnek hozzá. A csatolás-azonosítók egy szerkesztési alkalmon belül egyediek. A következő szerkesztési alkalommal a rendszer teljesen új azonosítókat rendel a csatolásokhoz. A frissített csatolás azonosítója nem változik és megtartja korábbi értékét, ha a csatolás a VISSZA parancs végrehajtása után helyreáll.

A rajzelem csatolás létrehozásához a következő attribútumokat kell megadni: csatolás útvonalnév, kulcsérték és rajzelemnév.

A megjeleníthető attribútum csatolás létrehozásához az alábbi attribútumokat kell megadni: csatolás útvonalnév, kulcsérték, megjeleníthető attribútum oszlop nevek és megjeleníthető attribútum paraméterek. A megjeleníthető attribútum paraméterek reprezentációjának meg kell felelnie a szövegobjektum követelményeinek. Az illesztési pont és az elforgatási szög szükséges a megjeleníthető attribútum csatolás létrehozásához. A megjeleníthető attribútum csatolás sikeres létrehozása után az ASE a megfelelő értéket rendeli a rajzelemnév attribútumhoz. A megjeleníthető attribútum csatolás létrehozása során meg lehet adni a megjeleníthető attribútum oszlop értékek attribútumát. Amennyiben ez az attribútum nincs megadva, akkor egy csillag szimbólum jelenik meg. A meglévő csatolás megjeleníthető attribútum paraméterek értékeinek és megjeleníthető attribútum oszlop értékek attribútumainak lekérdezésére nincs lehetőség.

Az ASE felület lehetővé teszi, hogy blokkokon belül hozzon létre vagy módosítson rajzelemekhez tartozó csatolásokat. Ebben az esetben az xref vagy blokknév attribútum taralmazza a blokknevet. A csatolás létrehozásakor az ASE felület automatikusan megvizsgálja, hogy a megadott rajzelem a blokkhoz tartozik-e, és a megfelelő blokknevet az attribútumhoz rendeli. Az ASE felület lehetővé teszi, hogy a csatolt rajzelemek nevének kicserélésével frissítse a csatolásokat.

Sikeres hívás esetén az újonnan létrehozott csatolás azonosítóját adja vissza. Máskülönben -1 a visszatérési érték.

Megjegyzés Mivel a függvény megváltoztathatja a rajz adatait, aktív párbeszédpanel esetén kerülje a használatát.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_linkget

A csatolás-azonosítóval meghatározott csatolás adattal tér vissza

(ase_linkget *link_data*)

Az **"ase_linkcreate"** függvénynél megtalálható a *link_data* argumentum leírása. Ha a csatolás nem létezik vagy valami nincs rendben, akkor az **ase_linkget** visszatérési értéke nil.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_linkremove

Megszünteti a csatolás-azonosítóval megadott csatolást

(ase_linkremove *link_id*)

A *link_id* argumentum egy valós érték, amely megadja a csatolást az xref-ből. A csatolt rajzelemnek léteznie kell, és frissíthetőnek kell lennie.

Sikeres működés esetén `T`, különben pedig `nil` a visszatérési érték.

Megjegyzés Mivel a függvény megváltoztathatja a rajz adatait, aktív párbeszédpanel esetén kerülje a használatát.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon bővebb információ az ASE argumentumokról és adattípusokról

Külsőleg definiált függvény

ase ARX alkalmazás

ase_linkupdate Frissíti a meglévő csatolást

(ase_linkupdate *link_data*)

Az **"ase_linkcreate"** függvénynél megtalálható a *link_data* argumentum leírása. Rajzelem csatolás esetén csak a rajzelemnevet és/vagy a kulcsértéket, illetve a csatolás útvonalnevet kell frissíteni. Megjeleníthető attribútum csatolás esetén csak a kulcsértéket vagy a megjeleníthető attribútum oszlop értékeket kell frissíteni. Más csatolás attribútumot figyelmen kívül hagy a függvény. A meglévő csatolás csatolt rajzelemének és az új megadott rajzelemnek létezniük kell és frissíthetőnek kell lenniük.

Sikeres hívás esetén `T`, máskülönben `nil` a visszatérési érték.

Megjegyzés Mivel a függvény megváltoztathatja a rajz adatait, aktív párbeszédpanel esetén kerülje a használatát.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lpcreate Új csatolás útvonalat hoz létre

(ase_lpcreate *table_path lp_name key*)

A *table_path* argumentum az adatbázis táblát, az *lp_name* pedig a csatolás útvonalnevet adja meg. A *key* argumentum egy lista, amely megadja a kulcsoszlop nevek listáját. A BIT, BIT VARYING és implementáció által definiált adattípusú oszlopnevek nem frissíthetők. A megfelelő környezetet csatolni kell, a táblának léteznie kell és elérhetőnek kell lennie. Akkor is `nil` értékkel tér vissza, ha az adott csatolás útvonalnév már létezik.

Sikeres hívás esetén `T`, máskülönben `nil` a visszatérési érték.

Megjegyzés Mivel a függvény megváltoztathatja a rajz adatait, aktív párbeszédpanel esetén kerülje a használatát.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lperase

A csatolás útvonalnév által megadott csatolás útvonalat törli

(ase_lperase *lp_name*)

Az *lp_name* argumentum egy karakterlánc, amely megadja a csatolás útvonalnevet. A függvény hatással lehet az aktuális beállításokra az által, hogy törli a csatolás útvonalnevet vagy az aktuális adatbázis-objektum bejegyzését.

Sikeres hívás esetén `T` értékkel tér vissza. Ha a csatolás útvonal nem frissíthető vagy ha valami nincs rendben, a visszatérési érték `nil` .

Megjegyzés Mivel a függvény megváltoztathatja a rajz adatait, aktív párbeszédpanel esetén kerülje a használatát.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lpisupdatable

Ellenőrzi, hogy a csatolás útvonal egészében frissíthető-e

(ase_lpisupdatable *lp_name*)

Megjegyzés A csatolás útvonal két esetben nem frissíthető: (1) ha nevének előtagjaként egy xref neve áll, (2) ha bármelyik csatolás nem frissíthető objektumra hivatkozik (egy zárt fólián vagy egy xref-en belül). Ha a csatolás útvonal frissíthető, a visszatérési érték `T`, különben `nil`.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lpkey

Az adott csatolás táblához tartozó kulcsoszlop nevekkel tér vissza

(ase_lpkey *lp_name*)

Az `lp_name` argumentum megadja a csatolás útvonalnevet. Sikeres hívás esetén oszlopneveket tartalmazó listát ad vissza. Máskülönben `nil` a visszatérési érték.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lplist

Csatolás útvonalnevek listájával tér vissza

(ase_lplist *[obj_path]*)

Az `obj_path` argumentum megadja az adatbázis-objektum hierarchiaszintjét. A függvény az adott adatbázis-objektum refrenciával összefüggő csatolás útvonalnevek listájával tér vissza. Amennyiben az `objpath` nincs megadva vagy egy üres karakterlánc, a függvény az összes létező csatolás útvonalnév listájával tér vissza. Sikeres hívás esetén egy karakterlánc listával tér vissza, máskülönben `nil` a visszatérési érték.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lppath

Adott csatolás útvonalhoz tartozó tábla útvonallal tér vissza

(ase_lppath *lp_name*)

Sikeres hívás esetén a tábla útvonallal tér vissza és `nil` a visszatérési érték, ha az adott csatolás útvonal nem létezik vagy valami nincs rendben.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lprename

Átnevezi a csatolás útvonalat

(ase_lprename *lp_oldname lp_newname*)

Az `lp_oldname` karakterlánc a régi csatolás útvonalnevet, az `lp_newname` karakterlánc pedig az új csatolás útvonalnevet adja meg. A függvény az aktuális adatbázis-objektum csatolás útvonal nevének megváltoztatásával hatással lehet az aktuális beállításokra.

Sikeres hívás esetén `T` értékkel tér vissza és `nil` a visszatérési érték, ha a csatolás útvonal nem frissíthető vagy valami baj van. Akkor is `nil` a visszatérési érték, ha az adott csatolás útvonalnév nem létezik.

Megjegyzés Mivel a függvény megváltoztathatja a rajz adatait, aktív párbeszédpanel esetén kerülje a használatát.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsadd

Csatolás-azonosítót ad a csatolás kiválasztáshoz

(ase_lsadd *link_sel link_id*)

Sikeres hívás esetén a visszatérési érték `T`, máskülönben `nil` .

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lscmp

Összehasonlítja a csatolás kiválasztás azonosítóit

```
(ase_lscmp link_sel1 link_sel2)
```

Hasonlóság esetén `T`, máskülönben `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lscopy

A csatolás kiválasztást egy újabba másolja

```
(ase_lscopy link_sel)
```

Sikeres hívás esetén `T`, máskülönben `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lscreate

Adott csatolásszűrő szerinti csatolás kiválasztást hoz létre

```
(ase_lscreate link_filter)
```

A `link_filter` argumentum a következő formában jelenik meg:

```
csatolás_attr1.attr_érték1 csatolás_attr2.attr_érték2 ...
```

A `link_filter` csatolás attribútumokat és azok értékeit tartalmazza párokban. A függvény az összes ilyen párhoz logikai AND műveletet rendel. A csatolás attribútum leírását az **"ase_linkcreate"** függvénynél találja meg. Amennyiben egyetlen csatolás attribútumpár sincs megadva, a visszaadott csatolás kiválasztás az összes létező csatolásreferenciát tartalmazza.

A következő csatolás attribútumok adhatók meg a csatolásszűrőben: csatolás-azonosító, csatolás típus kód, rajzelemnév, csatolás útvonalnév, kulcsérték, xref/blokk név. A szabvány csatolás attribútumokhoz az alábbi táblázatban megadott értékek adhatók még.

További link_filter értékek

Érték	*Leírás*
–1	Szövegfeltétel (szöveges kritérium)
–2	Kiválasztási halmaz neve (grafikus kritérium)
–3	Adatbázis-objektum útvonal

A csatolás kiválasztás a csatolások olyan azonosítóit tartalmazza, amelyek eleget tesznek a megadott WHERE SQL feltételnek, amennyiben a szövegfeltétel bővített attribútum adott. Amennyiben a kiválasztási halmaz neve adott, akkor a csatolások inicializálják, rajzelemekhez társul, és a kiválasztási halmaz tartalmazza a csatolás kiválasztást. Az adatbázis-objektum útvonal az adott adatbázis-objektummal összefüggő csatolás útvonalak csatolásainak, valamint az egész táblasorral összefüggő összes csatolásának kiválasztására szolgál.

A szövegfeltétel és kulcsérték attribútumnak követnie kell a csatolás útvonalnevet vagy az adatbázis-objektum útvonalat. Ebben az esetben az adatbázis-objektum attribútumnak tartalmaznia kell a táblára történő hivatkozást. Amennyiben az adott sor csatolás kiválasztásának kiválasztására használja a csatolásszűrőt (bármilyen csatolás útvonalnévvel, ha a csatolás útvonalnév nincs magadva a tábla referenciában), a kulcsértéknek tartalmaznia kell az összes sor értéket. Az összes csatolás megadásához az adatbázis objektum útvonalnak üres karakterláncnak kell lennie.

Minden létrehozott csatolás kiválasztást az **ase_lsfree** függvény segítségével kell felszabadítani, ha már nem használja őket. A csatolás kiválasztás minden egyes szerkesztési alkalommal egyedi.

Sikeres hívás esetén a csatolás kiválasztás azonosítóval tér vissza, különben nil a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsdel

Törli a csatolás pozíciót a csatolás kiválasztásból

(ase_lsdel *link_sel link_num*)

Sikeres hívás esetén a visszatérési érték `T`, máskülönben `nil` .

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsentsel

A csatolás kiválasztásban található csatolásokkal összefüggő objektumok halmazával (rajzelemekkel) tér vissza

(ase_lsentsel *link_sel*)

Ez a függvény nem dolgozza fel az xref/blokk hierarchiát. Az összes hivatkozott csatolásnak léteznie kell. Sikeres hívás esetén a csatolás kiválasztás nevével tér vissza, máskülönben `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lserase

Törli a csatolásokat és csatolás kiválasztást

(ase_lserase *link_sel*)

Az összes hivatkozott csatolásnak léteznie kell és frissíthetőnek kell lennie. Sikeres hívás estén `T`, hiba esetén `nil` a visszatérési érték.

Megjegyzés Mivel a függvény megváltoztathatja a rajz adatait, aktív párbeszédpanel esetén kerülje a használatát.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsfree

Felszabadítja a csatolás kiválasztást

(ase_lsfree *link_sel*)

Hiba esetén `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsget

A csatolás kiválasztásban szereplő adott pozíciójú csatolás-azonosítóval tér vissza

(ase_lsget *link_sel link_num*)

Hiba esetén `nil` a visszatérési érték.

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsintersect

A megadottakon kívül minden csatolás-azonosítót töröl egy csatolás kiválasztásból

(ase_lsintersect *link_sel link_sel_intersect*)

Törli a *link_sel* argumentum által megadott összes olyan csatolás-azonosítót, amely nem szerpel a *link_sel_intersect* által megadott csatolás kiválasztásban. Sikeres hívás esetén T, máskülönben nil a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsintersectfilter

A megadott csatolásszűrővel nem összefüggő azonosítókat törli a csatolás kiválasztásból

(ase_lsintersectfilter *link_filter*)

A *link_filter* argumentum párokban tartalmaz csatolás attribútumokat és azok értékeit (részletes leírás az **ase_lscreate** függvénynél található). Amennyiben egyetlen csatolás attribútumpár sincs megadva, a visszaadott csatolás kiválasztás az összes létező csatolásreferenciát tartalmazza.

Sikeres hívás esetén T, máskülönben nil a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsisupdatable

Ellenőrzi, hogy a csatolás kiválasztás összes csatolása frissíthető-e

(ase_lsisupdatable *link_sel*)

Egy csatolás kiválasztás akkor nem frissíthető, ha a hivatkozott rajzelem egy xref zárt fóliájához tartozik.

Amennyiben a csatolás kiválasztás frissíthető, a függvény `T` értékkel tér vissza, máskülönben `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lslpnames

A csatolás kiválasztás összefüggő csatolás útvonal neveinek listájával tér vissza

(ase_lslpnames *link_sel*)

Az összes hivatkozott csatolásnak léteznie kell. Sikeres hívás esetén karakterláncok listájával tér vissza, máskülönben `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsmemb

Ellenőrzi, hogy a csatolás-azonosító a csatolás kiválasztáshoz tartozik-e

(ase_lsmemb *link_sel link_id*)

Sikeres hívás esetén a megjelölt csatolás elemének sorszámával tér vissza, máskülönben `-1` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsqty

A csatolás kiválasztásban található csatolások számával tér vissza

(ase_lsqty *link_sel*)

Hiba esetén `nil` értékkel tér vissza.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lssubtract

Adott csatolás-azonosítókat töröl egy csatolás kiválasztásból

(ase_lssubtract *link_sel link_sel_subtract*)

Törli a `link_sel` által megadott összes csatolás-azonosítót, amelyek szerpelnek a `link_sel_subtract` által megadott csatolás kiválasztásban. Sikeres hívás esetén `T`, máskülönben `nil` a visszatérési érték .

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalamzás

ase_lsunite

Egy csatolás kiválasztást ad a másikhoz

(ase_lsunite *link_sel link_sel_union*)

A `link_sel_union` argumentummal megadott csatolás kiválasztást hozzáadja `link_sel` argumentummal megadotthoz. Sikeres hívás esetén `T`, máskülönben `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_lsxnames

A csatolás kiválasztás összefüggő xref vagy blokk neveinek listájával tér vissza

(ase_lsxnames *link_sel*)

Ez a függvény nem dolgozza fel az xref vagy a blokkok hierarchiáját. Az összes hivatkozott csatolásnak léteznie kell. Sikeres hívás esetén karakterláncok listájával tér vissza, máskülönben `nil` a visszatérési érték.

További tudnivalók

"Az ASE, az AutoLISP felhasználói felülete" című résznél, a 259. oldalon

Külsőleg definiált függvény

ase ARX alkalmazás

ase_version

Megjeleníti az aktuális ASE verziót

(ase_version)

Sikeres hívás esetén az ASE verziót jelző karakterlánccal tér vissza, máskülönben `nil` a visszatérési érték.

Külsőleg definiált függvény

ase ARX alkalmazás

ascii

Egy karakterlánc első karakterének ASCII kódjával (egy egész szám) tér vissza

(ascii *string*)

```
(ascii "A")          eredménye 65
(ascii "a")          eredménye 97
(ascii "BAL")        eredménye 66
```

A BASIC® nyelv **ASC** függvényéhez hasonló.

assoc

Egy elemet keres egy asszociációs listában és az asszociációs lista bejegyzésével tér vissza

(assoc *item alist*)

Az *alist* által megadott listában megkeresi az *item* elemet és annak a listában található bejegyzésével tér vissza. Amennyiben az **assoc** függvény az *alist* listában nem talál *item* nevű elemet, `nil` értékkel tér vissza.

Például az alábbi kód esetén:

```
(setq al '((név doboz) (szélesség 3) (méret 4.7263)
(mélység 5)))
```

ebből

```
(assoc 'méret al)    eredménye (MÉRET 4.7263)
(assoc 'súly al)     eredménye nil
```

Asszociációs listákat gyakran használnak olyan adatok tárolására, amelyeket valamilyen *kulcs* segítségével lehet elérni. A **subst** függvény kényelmes lehetőséget nyújt egy kulcshoz rendelt érték kicserélésére egy asszociációs listában.

atan

Egy szám árkusz tangensével tér vissza radiánban

(atan *num1 [num2]*)

Amennyiben Ön csak egy argumentumot ad meg az **atan** függvénynek, akkor az a *num1* argumentum árkusz tangensével tér vissza radiánban. Amennyibe Ön megadja a *num1* és a *num2* argumentumot is, akkor az **atan** függvény a *num1*/*num2* árkusz tangensével tér vissza radiánban. Amennyiben *num2* nulla, a függvény plusz vagy mínusz 1.570796 radián értékkel tér vissza (+90 fok vagy –90 fok), a *num1* előjelétől függően. A visszaadott szögek értéke –pi/2 és +pi/2 közti radián.

```
(atan 0.5)          eredménye 0.463648
(atan 1.0)          eredménye 0.785398
(atan -1.0)         eredménye -0.785398
(atan 2.0 3.0)      eredménye 0.588003
(atan 2.0 -3.0)     eredménye 2.55359
(atan 1.0 0.0)      eredménye 1.5708
```

Megjegyzés Az **angtos** függvény az **atan** függvény által visszaadott radián értéket konvertálja át egy karakterlánc-értékké.

```
(angtos (atan -1.0) 0 4)      eredménye "315.0000"
(angtos (atan 2.0 3.0) 0 4)   eredménye "33.6901"
(angtos (atan 2.0 -3.0) 0 4)  eredménye "146.3099"
(angtos (atan 1.0 0.0) 0 4)   eredménye "90.0000"
```

atof

Egy karakterlánc valós számmá konvertált értékével tér vissza

(atof *string*)

```
(atof "97.1")       eredménye 97.1
(atof "3")          eredménye 3.0
(atof "3.9")        eredménye 3.9
```

atoi

Egy karakterlánc egész számmá konvertált értékével tér vissza

(atoi *string*)

```
(atoi "97")          eredménye 97
(atoi "3")           eredménye 3
(atoi "3.9")         eredménye 3
```

atom

Ellenőrzi, hogy egy elem atom-e

(atom *item*)

A visszatérési érték `nil`, ha az *item* egy lista, máskülönben `T` értékkel tér vissza. Minden atomnak számít, ami nem lista.

Például, az alábbi értékadás esetén

```
(setq a '(x y z))
(setq b 'a)
```

ebből

```
(atom 'a)            eredménye T
(atom a)             eredménye nil
(atom 'b)            eredménye T
(atom b)             eredménye T
(atom '(a b c))      eredménye nil
```

Néhány LISP verzió eltér az **atom** fogalmának meghatározásában, ezért óvatosan használja a konvertált kódokat.

atoms-family

A pillanatnyilag definiált szimbólumok listájával tér vissza

(atoms-family *format [symlist]*)

A *format* argumentum egy egész szám, melynek értéke 0 vagy 1. Amennyiben a *format* értéke 0, akkor az **atoms-family** listaként adja vissza a szimbólumneveket. Amennyiben a *format* értéke 1, akkor a függvény karakterláncok listájaként adja vissza a szimbólumneveket. Az **atoms-family** függvény egy specifikus szimbólumnév-listát keres, amennyiben Ön megadja a *symlist* argumentumot is. A *symlist* argumentum egy karakterláncokból álló lista, amely megadja a szimbólumneveket. Az **atoms-family** függvény a format argumentum által megadott típusú listát ad vissza (szimbólum vagy karakterlánc),

és a definiált szimbólumneveket tartalmazza. A nem definiált szimbólumoknál `nil` értéket ad vissza.

`(atoms-family 0)` *Az aktuálisan definiált szimbólumok listáját adja vissza*

A következő kód ellenőrzi, hogy a `CAR`, `CDR`, és `XYZ` szimbólumokat definiálták-e korábban, és karakterláncokból álló listát ad vissza:

```
(atoms-family 1
  '("CAR" "CDR" "XYZ"))    eredménye    ("CAR" "CDR" nil)
```

Az előbbi visszatérési érték azt mutatja, hogy az `XYZ` szimbólum nem volt definiálva.

autoarxload

Parancsneveket definiál, amelyek egy asszociált ARX fájlt töltenek be

(autoarxload *filename cmdlist*)

A *filename* argumentum egy karakterlánc, amely meghatározza a betöltendő *.arx* fájlt, ha a felhasználó a *cmdlist* argumentum által megadott parancsok egyikét írja be a parancsprompt után. A *cmdlist* argumentumnak egy karakterláncokból álló listának kell lennie. Az **autoarxload** függvény `nil` értékel tér vissza.

A következő kód megadja, hogy a `C:ALK1`, `C:ALK2`, és `C:ALK3` függvények a *bonusalk.arx* fájlt töltsék be. Az első alkalommal, mikor az `ALK1`, `ALK2`, vagy `ALK3` parancsok valamelyikét a felhasználó beírja a parancsprompt után, az ARX alkalmazás betöltődik, és folytatódik a parancs végrehajtása.

```
(autoarxload "BONUSALK" '("ALK1" "ALK2" "ALK3"))
```

Külsőleg definiált függvény

acadr13.lsp AutoLISP alkalmazás

autoload

Parancsneveket definiál, amelyek egy asszociált AutoLISP fájlt töltenek be

(autoload *filename cmdlist*)

A *filename* argumentum egy karakterlánc, amely megadja a betöltendő *.lsp* fájlt, ha a felhasználó a *cmdlist* argumentum által megadott parancsok egyikét írja be a parancsprompt után. A *cmdlist*

argumentumnak egy karakterláncból álló listának kell lennie. Az **autoload** függvény `nil` értékkel tér vissza.

A következő kód megadja, hogy a `C:ALK1`, `C:ALK2`, és `C:ALK3` függvények a *bonusalk.lsp* fájlt töltsék be. Az első alkalommal, mikor az `ALK1`, `ALK2`, vagy `ALK3` parancsok valamelyikét a felhasználó beírja a parancsprompt után, az AutoLISP fájl betöltődik, és folytatódik a parancs végrehajtása.

```
(autoload "BONUSALK" '("ALK1" "ALK2" "ALK3"))
```

Külsőleg definiált függvény

acadr13.lsp AutoLISP alkalmazás

autoxload

Parancsneveket definiál, amelyek egy asszociált ADS alkalmazást töltenek be

(autoxload *filename cmdlist*)

A *filename* argumentum egy karakterlánc, amely megadja a betöltendő ADS alkalmazást, ha a felhasználó a *cmdlist* argumentum által megadott parancsok egyikét írja be a parancsprompt után. A *cmdlist* argumentumnak egy karakterláncból álló listának kell lennie. Az **autoload** függvény `nil` értékkel tér vissza.

A következő kód megadja, hogy a `C:ALK1`, `C:ALK2`, és `C:ALK3` függvények a *bonusalk* ADS alkalmazást töltsék be. Az első alkalommal, mikor az `ALK1`, `ALK2`, vagy `ALK3` parancsok valamelyikét a felhasználó beírja a parancsprompt után, az ADS alkalmazás betöltődik, és folytatódik a parancs végrehajtása.

```
(autoload "BONUSALK" '("ALK1" "ALK2" "ALK3"))
```

Külsőleg definiált függvény

acadr13.lsp AutoLISP alkalmazás

Boole

Általános bitenkénti Boolean függvényként szolgál

(Boole *func int1 int2 ...*)

A *func* argumentum egy 0 és 15 közötti egész szám, amely két változó lehetséges 16 Boolean függvényének egyikét jeleníti meg. Az egymás

utáni egész szám argumentumok bitenként (logikailag) kapcsolódnak össze a függvénynek és a következő igazságtáblának megfelelően.

Boolean igazságtábla

Int1	*Int2*	*Func bit*
0	0	8
0	1	4
1	0	2
1	1	1

Az `int1` minden egyes bitjét párba állítja az `int2` megfelelő bitjével. Ezzel az igazságtábla egy vízszintes sorát jelöli ki. Az eredményként kapott bit 0 vagy 1, annak a `func` bitnek az állásától függően, amely az igazságtábla megfelelő sorához tartozik.

Amennyiben a `func` megfelelő bitje be van állítva, az eredményként kapott bit 1; máskülönben az eredményként kapott bit 0. A `func` egyes értékei az AND, OR, XOR és NOT szabvány Boolean műveleteknek felelnek meg .

Boole függvény bit értékei

Függv	*Művelet*	*Eredmény bit 1 ha...*
1	AND	Mindkét bemenő bit 1
6	XOR	Csak az egyik bemenő bit 1
7	OR	Egyik vagy mindkét bemenő bit 1
8	NOR	Mindkét bemenő bit 0 (1-es komplemens)

Az alábbi utasítás a 12 és 5 értékek logikai AND műveletét adja meg:

```
(Boole 1 12 5)          eredménye 4
```

Hasonlóképpen, a következő a 6 és 5 értékek logikai XOR műveletét adja meg:

```
(Boole 6 6 5)           eredménye 3
```

A *func* más értékeit is használhatja olyan Boolean műveletek végrehajtására, amelyeknek nincs szabvány nevük. Például, ha a *func* értéke 4, az eredmény bitek értéke akkor lesz 1, ha a megfelelő bitek értéke az *int2* argumentumban 1 és az *int1* argumentumban 0. Így

```
(Boole 4 3 14)          eredménye 12
```

boundp

Ellenőrzi, hogy egy szimbólumnak van-e értéke

(boundp *sym*)

A visszatérési érték T, ha a *sym* szimbólumhoz redeltek valamilyen értéket. Amennyiben nincs érték rendelve a *sym* szimbólumhoz (vagy nil értéket rendeltek hozzá), a **boundp** nil értékkel tér vissza. Amennyiben a *sym* egy nem definiált szimbólum, akkor azt a függvény automatikusan létrehozza, és nil értéket rendel hozzá.

Például, ha a következőt adjuk meg

```
(setq a 2 b nil)
```

akkor

```
(boundp 'a)             eredménye T
(boundp 'b)             eredménye nil
```

Az **atoms-family** függvény alternatív módszert nyújt annak megállapítására, hogy a szimbólum létezik-e, anélkül, hogy automatikusan létrehozná azt.

car and cdr

Egy lista első elemével tér vissza, vagy egy olyan listával, amely az első elem kivételével tartalmazza az összes elemet

(car *list*) és **(cdr *list*)**

Amennyiben a *list* üres, a **car** visszatérési értéke nil.

```
(car '(a b c))          eredménye A
(car '((a b) c))        eredménye (A B)
(car '())             eredménye nil
```

Amennyiben a *list* üres, a **cdr** visszatérési értéke nil.

```
(cdr '(a b c))          eredménye (B C)
(cdr '((a b) c))        eredménye (C)
(cdr '())               eredménye nil
```

Amikor a *list* argumentumban a párokat pont választja el (részletesebb információ a **"cons"** függvénynél található), a **cdr** a

második elemmel tér vissza, anélkül, hogy csatolta volna azt egy listához.

```
(cdr '(a . b))                  eredménye B
(cdr '(1 . "Szöveg"))           eredménye "Szöveg"
```

Az AutoLISP maximum négy szint mélységig támogatja a **car** és **cdr** függvények összefűzését. Az alábbiak érvényes függvények:

caaaar	**cadaar**	**cdaaar**	**cddaar**
caaadr	**cadadr**	**cdaadr**	**cddadr**
caaar	**cadar**	**cdaar**	**cddar**
caadar	**caddar**	**cdadar**	**cdddar**
caaddr	**cadddr**	**cdaddr**	**cddddr**
caadr	**caddr**	**cdadr**	**cdddr**
caar	**cadr**	**cdar**	**cddr**

Ezek az összefűzések egymásba ágyazott **car** és **cdr** függvényhívásoknak felelnek meg. Minden **a** egy **car**, és minden ***d*** egy **cdr** függvényhívást jelent. Például:

```
(caar x)    egyenlő  (car (car x))
(cdar x)    egyenlő  (cdr (car x))
(cadar x)   egyenlő  (car (cdr (car x)))
(cadr x)    egyenlő  (car (cdr x))
(cddr x)    egyenlő  (cdr (cdr x))
(caddr x)   egyenlő  (car (cdr (cdr x)))
```

Az AutoLISP nyelvben a **cadr** függvényt gyakran használják egy két- vagy háromdimenziós pont *Y* koordinátájának (a két vagy három valós számból álló lista második elemének) megállapítására. Hasonlóan a **caddr** függvény egy háromdimenziós pont *Z* koordinátájának megállapítására használható. Például, ha az alábbiakat adjuk meg

```
(setq pt2 '(5.25 1.0))          egy kétdimenziós pont
(setq pt3 '(5.25 1.0 3.0))      egy háromdimenziós pont
```

akkor

```
(car pt2)                       eredménye 5.25
(cadr pt2)                      eredménye 1.0
(caddr pt2)                     eredménye nil
(car pt3)                       eredménye 5.25
(cadr pt3)                      eredménye 1.0
(caddr pt3)                     eredménye 3.0
```

chr

Egy ASCII karakterkódot képviselő egész számot egykarakteres karakterlánccá konvertál

(chr *integer*)

```
(chr 65)    eredménye "A"
(chr 66)    eredménye "B"
(chr 97)    eredménye "a"
```

A függvény a BASIC nyelv **chr$** függvényéhez hasonló.

client_data_tile

Alkalmazás által kezelt adatokat társít egy párbeszédpanel-zónához

(client_data_tile *key clientdata*)

A *key* argumentum olyan karakterlánc, amely egy zónát ad meg. A *key* argumentum különbséget tesz a kis- és nagybetűk között. Az adat olyan karakterlánc, amelyet a *clientdata* argumentum ad meg. Egy akciókifejezés vagy visszajelzési függvény a karakterláncra mint `$data` változóra hivatkozhat.

close

Lezár egy nyitott fájlt

(close *file-desc*)

A *file-desc* argumentum az **open** függvény által visszaadott fájlleíró. Egy **close** után a fájlleíró változatlan marad, de a továbbiakban már nem érvényes. A **close** függvény `nil` értékkel tér vissza.

Például, ha az `x` egy érvényes fájlleíró:

`(close x)` *lezárja az érintett fájlt és* `nil` *értékkel tér vissza*

command

Egy AutoCAD parancsot hajt végre

(command *[arguments]* ...)

Az *arguments* argumentum AutoCAD parancsokat és azok opcióit képviseli. A **command** függvény argumentumai lehetnek karakterláncok, valós számok, egész számok vagy pontok, attól függően, hogy mit vár a végrehajtandó parancs promptsora. Az üres karakterlánc (`""`) a

↵ billentyű lenyomásának felel meg. A **command** függvény argumentum nélküli meghívása a Ctrl+C billentyűk lenyomásának felel meg és megszakítja a legtöbb AutoCAD parancsot. A **command** függvény `nil` értékkel tér vissza.

A **command** függvény kiértékel minden argumentumot és az egymást követő promptokra válaszolva továbbítja őket az AutoCAD rendszernek. A függvény a parancsneveket karakterláncként, a kétdimenziós pontokat két valós számból álló listaként, a háromdimenziós pontokat három valós számból álló listaként fogadja. Az AutoCAD csak akkor ismeri fel a parancsneveket, ha parancspromptot ad ki.

A következő példa a `pt1` és `pt2` változókat egy-egy pont értékkel (1,1 és 1,5) teszi egyenlővé. Majd a **command** függvényt használja arra, hogy a kiadja a VONAL parancsot és megadja a két pontértéket.

```
(setq pt1 '(1 1) pt2 '(1 5))
(command "vonal" pt1 pt2 "")
```

Amennyiben alkalmazásához idegennyelvű AutoCAD verziót használ, akkor az eredeti, angol nyelvű parancsneveket egy aláhúzás karakterrel (_) kell kezdeni, hogy azt a rendszer le tudja fordítani. Amennyiben a pont előtagot használja (az átdefiniált parancsok használatának elkerülése végett), akkor a pont és aláhúzás karaktereket bármilyen sorrendben alkalmazhatja; a `"._line"` és `"_.line"` is érvényes.

A **command** függvényben végrehajtott parancsok nem jelennek meg a parncssorban, ha a CMDECHO rendszerváltozó (amely a **setvar** és **getvar** függvényekkel érhető el) értéke 0.

Megjegyzés A **getxxx** felhasználói bevitelre szolgáló függvények (**getangle**, **getstring**, **getint**, **getpoint**, és így tovább) nem használhatók a **command** függvényen belül. Egy erre irányuló kísérlet a következő üzenetet eredményezi és befejezi a függvény működését.

hiba: az AutoCAD visszautasította a függvényt

Amennyiben felhasználói adatbevitel szükséges, hívja meg a **getxxx** függvényeket korábban, vagy illessze be őket az egymást követő **command** függvények közé.

Az olyan AutoCAD parancsok esetén, amelyeknél egy objektum kiválasztására van szükség (mint a MEGTÖR és METSZ parancsok), egy pont helyett az **entsel** függvény által visszaadott listát használhatja az objektum kiválasztására. További információ a "Kijelölt pontok átadása az AutoCAD parancsoknak" című résznél, a 226. oldalon.

Megjegyzés Az AutoCAD DSZÖVEG és SKICC parancsai közvetlenül olvassák a billentyűzetet, illetve a digitalizálót, ezért nem használhatóak az AutoLISP **command** függvénnyel. Amennyiben a FKÖNYV parancsot használja a **command** függvénnyel, akkor annak az utolsó függvény hívásnak kell lennie az AutoLISP rutinban.

A PAUSE szimbólum használata

Amennyiben egy AutoCAD parancs aktív és az előre definiált PAUSE szimbólummal találkozik, mint a **command** függvény a argumentumával, akkor a **command** függvény felfüggeszti működését, hogy közvetlen felhasználói adatbevitelt tegyen lehetővé.

A PAUSE szimbólumot egy egyetlen visszahajló törtjelet tartalmazó karakterláncként definiálták. Használhatja a visszahajló törtjelet a PAUSE szimbólum helyet. Azonban, ha a **command** függvényt egy menüpontból hívja meg, a visszahajló törtjel felfüggeszti a menüpont beolvasását, amely az AutoLISP kifejezés részleges kiértékelését eredményezi. A szünet mechanizmusnak esetleg más értéke lesz az AutoLISP következő verzióiban, ezért ajánlott a PAUSE szimbólum használata az explicit visszahajló törtjel helyett.

Amikor egy karakterláncban visszahajló törtjelet (\) kíván használni, akkor azt meg kell előznie egy másik visszahajló törtjelnek (\\).

Amennyiben egy parancs a PAUSE szimbólummal találkozik, mikor egy karakterlánc vagy egy attribútumérték bevitelére vár, az AutoCAD csak abban az esetben vár bevitelre, ha a TEXTEVAL rendszerváltozó értéke nem nulla. Máskülönben az AutoCAD nem vár felhasználói adatbevitelre, hanem a PAUSE szimbólum értékét (egyetlen visszahajló törtjel) használja.

Amikor a **command** függvény felhasználói adatbevitelre vár, akkor a függvényt aktívnak tekinti a rendszer, így a felhasználó nem írhat be újabb AutoLISP kifejezést.

Következzék egy példa a PAUSE szimbólum használatára (a ÚJ_FÓL fóliának léteznie kell a rajzon mielőtt a programot kipróbálná):

```
(setq blk "SAJÁT_BLOKKOM")
(setq régi_fól (getvar "clayer"))
(command "fólia" "beá" "ÚJ_FÓL " "")
(command "beill" blk pause "" "" pause)
(command "fólia" "beá" régi_fól "")
```

A fenti programrészlet aktuális fóliának a ÚJ_FÓL fóliát állítja be, vár, amíg a felhasználó kiválaszt egy beillesztési pontot a SAJÁT_BLOKKOM nevű blokk számára (melyet 1 értékű méretarány-tényezővel illeszt be) és ismét vár, míg a felhasználó egy forgatási szöget ki nem jelöl. Azután aktuális fóliának az eredeti fóliát állítja be.

Amennyiben a **command** függvény a KIJELÖL parancsot a PAUSE szimbólummal hívja meg és, ha egy PICKFIRST halmaz aktív, akkor a KIJELÖL parancs nem vár felhasználói bevitelre, hanem a PICKFIRST halmaz lesz a bevitt érték.

Figyelmeztetés A Méret prompt Sugár és Átmérő alparancsai bizonyos helyzetekben újabb promptot eredményeznek. Ez hibát okozhat a 11-es verzió előtt írt AutoLISP programokban, melyek használják ezeket a parancsokat.

További tudnivalók

"Parancsok meghívása a command függvénnyel" című részнél, a 224. oldalon további információ található a **command** függvényről

cond

Az AutoLISP elsődleges feltételes függvényeként szolgál

(cond (*test1 result1* ...) ...)

A **cond** függvény tetszőleges számú listát fogad el argumentumként. Kiértékeli minden lista első elemét (a megadott sorrendben), amíg ezen elemek valamelyike nem `nil` értékkel tér vissza. Majd azokat a kifejezéseket értékeli ki, amelyek a sikeres teszt kifejezést követik, és az allista utolsó kifejezésének értékével tér vissza. Amennyiben csak egy kifejezés található az allistában (vagyis hiányzik a *result* kifejezés), akkor a függvény a *test* kifejezés értékével tér vissza.

A következő példa a **cond** függvényt használja az abszolút érték kiszámítására:

```
(cond
  ((minusp a) (- a))
  (t a)
)
```

Amennyiben a változó értéke –10, akkor a visszatérési érték 10.

A fenti példából látható, hogy a **cond** függvényt *esetválasztásos* (case) típusú függvényként lehet használni. Gyakran a `T` értéket használják az utolsó (alapértelmezett) *test* kifejezésként. Következzék itt még egy egyszerű példa. Az `s` változóban a felhasználó által adott válasz található. A függvény teszteli a választ és 1 értékkel tér vissza, ha az `I` vagy `i`, illetve 0 értékkel, ha az `N` vagy `n`. Máskülönben `nil` a visszatérési érték.

```
(cond
  ((= s "I") 1)
  ((= s "i") 1)
  ((= s "N") 0)
  ((= s "n") 0)
  (t nil)
)
```

cons

Legegyszerűbb listakészítő függvény

(cons *new-first-element list*)

A **cons** függvény argumentuma egy elem (*new-first-element*) és egy *list* nevű lista. A függvény az elemet a lista elejére teszi és a listával tér vissza. Az első elem lehet atom vagy lista.

```
(cons 'a '(b c d))          eredménye  (A B C D)
(cons '(a) '(b c d))        eredménye  ((A) B C D)
```

A **cons** függvény atomot is elfogad a *list* argumentum helyén, így *kapcsolt értékpár* néven ismert struktúrát épít. Mikor az AutoLISP kapcsolt értékpárt jelenít meg az első és második elem közé pontot tesz. A kapcsolt értékpár második atomjának visszaadására használhatja a **cdr** függvényt.

```
(cons 'a 2)                 eredménye  (A . 2)
(car (cons 'a 2))           eredménye  A
(cdr (cons 'a 2))           eredménye  2
```

Egy kapcsolt értékpár nem más, mint egy lista. Néhány függvény, amely közönséges listákat kezel, nem fogadja el argumentumként .

cos

Egy radiánban kifejezett szög koszinuszával tér vissza

(cos *angle*)

```
(cos 0.0)                   eredménye   1.0
(cos pi)                    eredménye  -1.0
```

cvunit

Egyik mértékegységben mért értéket konvertál át egy másik mértékegységbe

(cvunit *value from to*)

A *value* argumentum az a numerikus érték, amelyet konvertálni kíván. Lehet két vagy három számból álló lista is (két- vagy háromdimenziós pont). A *from* argumentum az a mértékegység, amelyből, a *to* pedig, amelyikbe konvertál a függvény. A *from* és *to* argumentumok bármely, az *acad.unt* fájlban megtalálható mértékegységet megnevezhetnek.

Sikeres hívás esetén a **cvunit** a konvertált értékkel tér vissza. Amennyiben valamelyik mértékegység neve ismeretlen (nem található az *acad.unt* fájlban), vagy a két mértékegység nem kompatibilis egymással (mint például a gramm és az év), a **cvunit** visszatérési értéke `nil`.

```
(cvunit 1 "perc" "másodperc")          eredménye 60.0
(cvunit 1 "gallon" "méter")            eredménye nil
(cvunit 1.0 "hüvelyk" "cm")            eredménye 2.54
(cvunit 1.0 "hold" "négyzet yard")     eredménye 4840.0
(cvunit '(1.0 2.5) "láb" "hüvelyk")    eredménye (12.0 30.0)
(cvunit '(1 2 3) "láb" "hüvelyk")      eredménye (12.0 24.0
                                                 36.0)
```

Megjegyzés Amennyiben több értéket kíván hasonló módon konvertálni, hatékonyabb, ha az 1.0 értéket konvertálja egyszer, majd az eredményt arányossági tényezőként használja fel a saját függvényében vagy a további számításokban. Ezt alkalmazhatja az összes előre definiált mértékegység esetében, kivéve a hőmérsékletet, ahol eltolásra is szükség van.

További tudnivalók

"Mértékegységek átalakítása" című résznél, a 248. oldalon

defun

Egy függvényt definiál

(defun *sym argument-list expr ...*)

A **defun** függvény egy *sym* nevű függvényt definiál (a függvény nevét automatikusan idézőjelbe teszi). A függvény nevét egy argumentumlista követi (el is maradhat), majd egy törtjel után egy vagy több, a függvényhez tartozó lokális szimbólumot lehet megadni. A törtjelet legalább egy szóköznek el kell választani az első lokális szimbólumtól és az utolsó argumentumtól, ha van ilyen. Amennyiben nem ad meg egyetlen argumentumot vagy lokális szimbólumot sem, akkor egy üres kerek zárójelpárt kell a függvény neve után írnia.

A következő *argument-list* példák érvényes és érvénytelen értékeket adnak meg:

```
(defun sajátfüggv (x y) ...)          Két argumentuma van a függvénynek
(defun sajátfüggv (/ a b) ...)        Két lokális szibóluma van a függvénynek
(defun sajátfüggv (x / temp) ...)     Egy argumentum, egy lokális szimbólum
(defun sajátfüggv () ...)             Nincs sem argument, sem lokális szimbólum
```

Egy függvényt nem definiálhat több azonos nevű argumentummal. Ugyanakkor létezhet olyan argumentum, amely azonos néven definiál két lokális változót vagy az egyik argumentumot:

```
(defun fubar (a a / b) ...)       Nem érvényes
(defun fubar (a b / a a b) ...)   Érvényes
```

Amennyiben az argumentum/szimbólum lista azonos bejegyzéseket tartalmaz, akkor a függvény mindegyik név első előfordulását használja, és a többi előfordulást figyelmen kívül hagyja.

Az argumentumlistát és lokális szimbólumokat követő egy vagy több kifejezés akkor kerül kiértékelésre, amikor a függvényt meghívják.

A **defun** függvény a definiált függvény nevével tér vissza. Amikor a definiált függvényt meghívják, akkor az argumentumait kiértékeli és a az argumentum szimbólumokhoz rendeli azokat. A függvényen belül használhatja a lokális szimbólumokat anélkül, hogy külsőbb szinteken megváltoztatná azok kötődéseit. A függvény az utoljára kiértékelt kifejezés értékével tér vissza. Az összes korábbi kifejezéseknek csak mellékhatásuk van.

A következő példa új függvényeket definiál a **defun** függvény segítségével, és bemutatja azok visszatérési értékeit:

```
(defun add10 (x)
  (+ 10 x)
)                                 eredménye  ADD10
(add10 5)                         eredménye  15
(add10 -7.4)                      eredménye  2.6
```

és

```
(defun pontok (x y / temp)
  (setq temp (strcat x "..."))
  (strcat temp y)
)                                 eredménye  PONTOK
(pontok "a" "b")                  eredménye  "a...b"
(pontok "attól" "addig")          eredménye  "attól...addig"
```

Figyelmeztetés Ne használjon beépített függvény- vagy szimbólumneveket a *sym* argumentumban. Ez hozzáférhetetlenné teszi a beépített függvényt. A beépített és korábban definiált függvények kilistázására használja az **"atoms-family"** függvényt.

További tudnivalók

"Szimbólum- és függvénykezelés" című résznél, a 210. oldalon.

dictnext

Megkeresi egy szótárban a következő tételt

(dictnext *ename symbol [rewind]*)

Az *ename* argumentum egy rajzelemnév, amely megad egy keresni kívánt szótárobjektumot. A *symbol* argumentum egy karakterlánc, amely magadja a tételt a szótárban. Érvényes szimbólumnevek az ACAD_MLINESTYLE és ACAD_GROUP.

Amikor a **dictnext** függvényt többször használja egymás után, akkor az minden egyes alkalommal a szótár következő bejegyzésével tér vissza. A **dictsearch** függvény egy bejegyzést ad meg visszatérési értéknek. Amennyiben a *rewind* argumentum értéke nem nil, a szótár elejére áll a függvény, és az első bejegyzést adja vissza. Amikor nincs több bejegyzés a szótárban, akkor a visszatérési érték nil. Törölt bejegyzéseket nem ad vissza a függvény.

További információ található a fő szótár elemnévről a **"namedobjdict"** függvény leírásánál.

Megjegyzés Amennyiben már elkezdett egy szótár tartalmában lépkedni és egy másik szótár nevet ad meg a **dictnext** függvénynek, akkor az eredeti szótárbeli pozíció elvész. A függvény csak az éppen aktív szótárbeli pozíciót tartja nyilván.

Amennyiben a függvény megtalált egy tételt, akkor egy DXF típuskódok és értékek kapcsolt párjainak listájával tér vissza.

dictsearch

Egy tételt keres egy szótárban

(dictsearch *ename symbol [setnext]*)

Az *ename* argumentum egy rajzelemnév, amely a keresni kívánt szótárobjektumot adja meg. A *symbol* argumentum egy karakterlánc, amely magadja a tételt a szótárban. Érvényes szimbólumnevek az ACAD_MLINESTYLE és az ACAD_GROUP.

Amennyiben a **dictsearch** megtalálja az adott tételt, akkor a **dictnext** függvénynél leírt formában tér azzal vissza. Amennyiben nem talál ilyen tételt, akkor a visszatérési érték `nil`.

Normális esetben a **dictsearch** függvénynek nincs hatása **dictnext** által visszaadott tételek sorrendjére. Azonban, ha a **dictsearch** hívása sikeres volt és a *setnext* argumentum adott és értéke nem `nil`, akkor a tételszámlálót úgy állítja be, hogy a következő **dictnext** hívás a **dictsearch** által visszaadott érték utáni tétellel tér vissza.

További információ található a fő szótár elemnévről a **"namedobjdict"** függvény leírásánál.

dimx_tile és dimy_tile

Egy zóna méreteit adja vissza párbeszédpanel-egységekben

(dimx_tile *key*) és (dimy_tile *key*)

A **dimx_tile** függvény a zóna szélességével, a **dimy_tile** függvény pedig a magasságával tér vissza. A *key* argumentum mindkét függvény esetében egy karakterlánc, amely megadja a zónát. A *key* argumentum megkülönbözteti a kis- és nagybetűket.

A függvények a zónában megadható lehető legnagyobb értékű koordinátákkal térnek vissza; mivel a koordináták számozása nullától kezdődik, ezért ezek a függvények ténylegesen a teljes *X*, illetve *Y* irányú kiterjedésnél eggyel alacsonyabb értéket adnak vissza (*X*–1 és *Y*–1). A **dimx_tile** és **dimy_tile** függvények a **vector_image**, **fill_image** és **slide_image** függvények használatához nyújtanak segítséget, amelyek abszolút zóna koordináták megadását kívánják meg.

distance

Két pont közti háromdimenziós távolsággal tér vissza

(distance *pt1 pt2*)

`(distance '(1.0 2.5 3.0) '(7.7 2.5 3.0))` *eredménye* `6.7`
`(distance '(1.0 2.0 0.5) '(3.0 4.0 0.5))` *eredménye* `2.82843`

Amennyiben egyik vagy mindkét pont kétdimenziós, a **distance** figyelmen kívül hagyja bármely háromdimenziós pont *Z* koordinátáját és a pontok aktuális szerkesztősíkra vetített kétdimenziós távolságával tér vissza.

További tudnivalók

"Geometriai segédfüggvények" című résznél, a 238. oldalon

distof

Egy (lebegőpontos) valós számot ábrázoló karakterláncot konvertál át valós értékké

(distof *string [mode]*)

A *mode* argumentum megadja a karakterlánc formátumához használt mértékegységet. Az értéknek meg kell felelnie az AutoCAD LUNITS nevű renszerváltozójának megengedett értékeivel. Amennyiben a *mode* argumentum nincs megadva, a **distof** függvény az LUNITS aktuális értékét használja .

Lineáris mértékegységek értékei

Mode értéke	*A string formátuma*
1	Tudományos
2	Decimális
3	Mérnöki (láb és tizedes hüvelyk)
4	Építészeti (láb és hüvelyk törtrésze)
5	Tört

A *string* argumentumnak olyan karakterláncnak kell lennie, amelyet a **distof** függvény helyesen tud átalakítani a *mode* argumentum által megadott mértékegységre. Formátuma megegyezhet az **rtos** függvény által visszaadott formátummal, vagy az AutoCAD által megengedett, billentyűzetről bevitt formával. A **distof** és **rtos** függvények egymás ellentettjei. Amennyiben egy **rtos** által visszaadott karakterláncot ad meg a **distof** függvénynek, az garantáltan érvényes értékkel tér vissza, és fordítva (feltéve, hogy mode értékek megegyeznek).

Megjegyzés A **distof** függvény a 3-as és 4-es módot azonos módon kezeli. Vagyis, ha a *mode* értéke 3 (mérnöki) vagy 4 (építészeti), és a *string* a két forma közül ***bármelyik***, akkor a **distof** a helyes valós értéket adja vissza.

Amennyiben a **distof** hívása sikeres, akkor valós számmal tér vissza, máskülönben `nil` a visszatérési érték.

done_dialog

Lezárja a párbeszédpanelt

(done_dialog *[status]*)

A **done_dialog** függvényt egy akciókifejezésen vagy visszajelzési függvényen belül kell hívni (további információ az **"action_tile"** függvénynél).

Amennyiben megadja az opcionális *status* argumentumot, akkor annak egy pozitív egész számnak kell lennie. A **start_dialog** ezzel tér vissza, az OK vagy Elvet nyomógombnak megfelelő 1, illetve 0 helyett. Bármilyen 1-nél nagyobb *status* érték jelentése az alkalmazástól függ.

A **done_dialog** függvény egy kétdimenziós pontlistával tér vissza, amely a párbeszédpanelnek abban az időponban érvényes (*X*,*Y*) pozíciója, amikor a felhasználó kilép belőle. Ezt a pontot egy későbbi **new_dialog** hívásnak átadva a párbeszédpanelt a felhasználó által kiválasztott helyen nyithatja meg újra.

Megjegyzés Amennyiben egy olyan nyomógombhoz rendel visszajelzést, amelynek key attribútuma "accept" vagy "cancel" (általában ez az OK, illetve az Elvet nyomógomb), akkor a visszajelzésnek explicit módon meg kell hívnia a **done_dialog** függvényt. Ellenkező esetben a felhasználó "bennragadhat" a párbeszédpanelben. Amennyiben nem biztosít explicit visszajelzést ezekhez a nyomógombokhoz, hanem a szabvány kilépő nyomógombokat használja, akkor az AutoCAD automatikusan kezeli őket. Ezenkívül az "accept" nyomó-gombhoz tartozó explicit AutoLISP akciónak a *status* argumentum értékéül 1-et (vagy az alkalmazásban definiált egyéb értéket) kell megadnia; máskülönben a **start_dialog** függvény az alapértelmezés szerinti értékkel, vagyis 0-val fog visszatérni, amiből úgy tűnhet, mintha a párbeszédpanel el lenne vetve.

end_image

Befejezi az éppen aktív párbeszédpanel-ábra szerkesztését

(end_image)

Ez a függvény a **start_image** ellentettje.

end_list

Befejezi az éppen aktív párbeszédpanel-listával kapcsolatos műveleteket

(end_list)

Ez a függvény a **start_list** ellentettje.

entdel

Objektumokat (rajzelemeket) töröl vagy korábban törölt objektumokat állít vissza

(entdel *ename*)

Az *ename* által megadott rajzelemet törli, ha az megtalálható a rajzon. Az **entdel** függvény viszaállítja a rajzelemet, ha azt korábban még az aktuális szerkesztési alkalmon belül törölték. Amikor kilép a rajzból, a rendszer eltávolítja a törölt rajzelemet a rajzról. Az **entdel** függvény egyaránt képes törölni grafikus és nem grafikus rajzelemeket.

`(setq e1 (entnext))`	*A rajzon található első rajzelem nevét tölti az* `e1` *változóba*
`(entdel e1)`	*Törli az* `e1` *rajzelemet*
`(entdel e1)`	*Visszaállítja a törölt* `e1` *rajzelemet*

Az **entdel** függvény csak fő rajzelemeknél működik. Attribútumokat és vonallánc-csomópontokat nem tud a szülő rajzelemektől függetlenül törölni. A **command** függvény segítségével alárendelt rajzelemek módosítására használhatja az ATTEDIT, illetve a VLEDIT parancsokat.

Blokkon belül nem tud rajzelemeket törölni. Azonban átdefiniálhatja a blokkot az **entmake** függvény segítségével, a törölni kívánt rajzelem kivételével.

entget

Egy objektum (rajzelem) definíciós adatait adja vissza

(entget *ename [applist]*)

Az **entget** függvény az *ename* nevű rajzelem definíciós adatait adja vissza. Ez vonatkozik a grafikus és nem grafikus rajzelemekre is. Amennyiben megadja az *applist* opcionális, bejegyzett alkalmazásnevek listáját, akkor az **entget** függvény az adott alkalmazáshoz tartozó bővített adatokat is visszaadja.

Az **entget** által visszaadott adatok asszociációs listaként vannak kódolva, amelyből az **assoc** függvény segítségével tételeket különíthet el. A listában található objektumok rajzelem-adatainak minden egyes részéhez AutoCAD DXF csoportkód van rendelve.

Tegyük fel, hogy az utoljára létrehozott objektum a rajzon egy vonal az (1,2) ponttól a (6,5) pontig. Az **entlast** függvénnyel visszakeresheti az utolsó rajzelem nevét, és átadhatja azt az **entget** függvénynek.

```
(entget (entlast))
```

Ez a következőket adhatja vissza:

```
((-1 . <Elem neve: 60000014>)      Rajzelem neve
  (0 . "LINE")                     Objektum típusa
  (8 . "0")                        Fólia
  (10 1.0 2.0 0.0)                 Indulópont
  (11 6.0 6.0 0.0)                 Végpont
)
```

Az AutoLISP által használt DXF csoportkódok egy kicsit eltérnek a DXF fájlban található csoportkódoktól. Az AutoLISP DXF csoportkódokat az "Alkalmazás-specifikus DXF csoportkódok" című, 16. fejezet tárgyalja.

Mint a DXF esetében is, a rajzelem-fejléc tételeket (szín, vonaltípus, vastagság, "attribútumok következnek" flag és az elemsorszám) csak akkor exportálja, ha értékük nem alapértelmezett. A DXF fájllal ellentétben az opcionális rajzelem definíciós mezőket akkor is exportálja, ha alapértelmezett értékük van, és akkor is, ha nem. Ez leegyszerűsíti a feldolgozást. A programok mindig feltételezhetik, hogy ezek a mezők a rajtuk végzett műveletek általános algoritmusai számára mindig jelen vannak. Szintén a DXF fájltól eltérően az összetartozó *X*, *Y* és *Z* koordináták egy pontlistába vannak csoportosítva (10 1.0 2.0 3.0), ahelyett, hogy különálló 10-es, 20-as és 30-as csoportként jelennének meg.

A –1-es tétel a lista elején a rajzelem nevét tartalmazza. Az értékeket tartalmazó, különálló kapcsolt párok az **assoc** függvény segítségével kiemelhetők, ha a **cdr** függvényt használjuk értékeik szétválasztására.

A pontok alárendelt listái a többitől eltérően nem kapcsolt értékpárok. A konvenció az, hogy a **cdr** függvény az alárendelt lista csoportértékét adja vissza. Mivel egy pont egy két vagy három valós számból álló lista, az egész csoport egy három (vagy négy) elemű lista. A csoporton végzett **cdr** művelet a pontot jelentő listát adja vissza, tehát megmaradt az a konvenció, hogy a **cdr** függvény mindig az értéket adja vissza.

Az ilyen elemlistát feldolgozó függvények írásánál, bizonyosodjon meg róla, hogy azok nem érzékenyek az alárendelt listák sorrendjére. Használja erre az **assoc** függvényt. A rajzelem nevét tartalmazó –1-es csoport az elemlista elfogadása érdekében megenged módosító műveleteket, és szükségtelenné teszi a rajzelemnév hasonló szerkezetben való tárolását. Egy vonallánc végén álló seqend rajzelem vagy egy attribútumhalmaz egy –2-es csoportot tartalmaz. Az ezeken elvégzett **cdr** művelet a rajzelem fejlécének rajzelem nevét adja vissza. Ez lehetővé teszi, hogy megtalálja a fejlécet, úgy hogy egy alárendelt

rajzelemtől előre lépked a seqend rajzelemig, hogy aztán a –2-es csoporton, mint a rajzelem nevén a **cdr** műveletet alkalmazva megkapja a hozzátartozó fő rajzelemet.

Figyelmeztetés A csomópontokon végrehajtott **entget** művelet előtt ki kell olvasnia vagy át kell írnia a vonallánc-rajzelem fejlécét. Amennyiben az utoljára feldolgozott vonallánc-rajzelem különbözik attól, amelyikhez a csomópont tartozik, akkor a szélességre vonatkozó információ (a 40-es és 41-es csoport) elveszhet.

Egy objektumhoz tartozó összes pont az objektum Objektum koordináta-rendszerében (OKR) van kifejezve . A pont, vonal, háromdimenziós vonal, háromdimenziós felület, háromdimenziós vonallánc, háromdimenziós soklapháló és méretező objektumok esetében az OKR megfelel Világ koordináta-rendszernek (az objektum pontjai világpontok). Az összes többi objektum esetében az OKR a VKR-ből és az objektum kihúzási irányából (210-es csoportjából) származtatható. A nem VKR-ben rajzolt objektumokkal történő munka esetén, szüksége lehet a pontok VKR-be vagy az aktuális Felhasználói koordináta-rendszerbe való konvertálására a **trans** függvény segítségével.

entlast

Az utolsó nem törölt fő objektum (rajzelem) nevét adja vissza a rajzon

(entlast)

Az **entlast** függvényt gyakran használják **command** függvénnyel létrehozott rajzelem nevének lekérésére. A rajzelemnek nem kell a képernyőn vagy a kiolvasztott fólián lennie.

```
(setq e1 (entlast))
(setq e2 (entnext e1))
```

Az e1-*be tölti az utolsó rajzelem nevét a rajzon*

e2-*be nil értéket (vagy egy attribútum illetve csomópont alárendelt rajzelemnevet) tölt*

Amennyiben az Ön alkalmazásának szüksége van az utolsó nem törölt rajzelem (fő *vagy* alárendelt rajzelem) nevére, akkor definiáljon az alábbihoz hasonló függvényt és azt hívja meg az **entlast** helyett.

```
(defun lastent (/ a b)
  (if (setq a (entlast))          Az utolsó fő rajzelemet kérdezi le
    (while (setq b (entnext a))   Ha alárendelt rajzelemek követik,
                                  addig marad a ciklusban, míg
      (setq a b)                  azok el nem fogynak
    )
  )
  a                               Az utolsó fő vagy alárendelt
)                                 rajzelemet adja vissza
```

entmake

Egy új objektumot (rajzelemet) hoz létre a rajzon

(entmake ***[elist]*****)**

Az *elist* argumentumnak az **entget** függvény által visszaadottal megegyező formátumú rajzelem-definíciós adatlistának kell lennie. Az *elist* argumentumnak tartalmaznia kell a rajzelem definiálásához szükséges összes információt. Az **entmake** függvény egyaránt definiálhat grafikus és nem grafikus rajzelemet. Amennyiben bármelyik szükséges adat elmarad, akkor az **entmake** nil értékkel tér vissza, és a rajzelemet elveti. Amennyiben Ön opcionális adatot (például fóliát) hagy el, akkor az **entmake** az alapértelmezés szerinti értéket használja.

Amennyiben az **entmake** sikeresen létrehoz egy rajzelemet, akkor a rajzelem definíciós adatlistájával tér vissza. Ha az **entmake** nem tudja létrehozni a rajzelemet, akkor nil értékkel tér vissza.

Egy új rajzelem létrehozásának egyik módszere, hogy az **entget** függvénnyel lekérdezi egy rajzelem definíciós adatait, módosítja azokat és átadja a módosított adatokat az **entmake** függvénynek. Egy új rajzelem létrehozása előtt az **entmake** ellenőrzi a megadott fólianév, vonaltípusnév és szín érvényességét. Ha új fólianevet ad meg, az **entmake** automatikusan létrehozza az új fóliát. Az **entmake** függvény ellenőrzi a blokkneveket, méretezési stílus neveket, szövegstílus neveket és alakneveket is, ha a rajzelem típusának szüksége van rájuk .

A rajzelem típusának (például CIRCLE vagy LINE) az *elist* első vagy második mezőjének kell lennie. Amennyiben ez a típus a második mező, akkor csak a rajzelem neve előzheti meg. Ez az **entget** által visszaadott formátum. Ilyen esetekben figyelmen kívül hagyja a rajzelem nevét, mikor új rajzelemet hoz létre. Amennyiben az *elist* egy elemsorszámot tartalmaz, akkor azt is figyelmen kívül hagyja.

A következő programrészlet egy vörös kört hoz létre (4,4) középponttal és 1-es sugárral. Az opcionális fólia- és vonaltípus-mezők elmaradtak, ezért alapértelmezett értékeket vesznek fel.

```
(entmake
  '((0 . "KÖR")          Rajzelem típusa
    (62 . 1)             Szín
    (10 4.0 4.0 0.0)     Középpont
    (40 . 1.0)           Sugár
  )
)
```

Megjegyzés A fagyasztott fólián létrehozott objektumokat csak a fólia kiolvasztása után generálja újra a rendszer.

Összetett rajzelemek

Egy összetett rajzelemet (egy blokkdefiníciót, egy vonalláncot vagy egy attribútumokat tartalmazó blokkhivatkozást) több **entmake** hívással, az alárendelt rajzelemek (attribútumok vagy csomópomtok) definiálásával lehet létrehozni. Amikor az **entmake** felismeri, hogy összetett rajzelem létrehozása van folyamatban, akkor egy ideiglenes fájlt hoz létre a definíciós adatok összegyűjtésére. Minden egyes **entmake** hívásnál ellenőrzi az ideiglenes fájl meglétét. Ha létezik, akkor az új adatokat hozzáfűzi. Amikor az összetett rajzelem definiálása teljessé válik (a megfelelő seqend vagy endblk rajzelem hozzáfűzésével), akkor a megadott adatokat újra ellenőrzi, és az összetett rajzelemet hozzáadja a rajzhoz. Egy blokkdefiníció végén (egy endblk **entmake** művelete) a blokk nevével tér vissza a normális esetben visszaadott rajzelem adatainak listája helyett.

Megjegyzés Nézetablak objektumot nem hozhat létre az **entmake** függvénnyel

Ha a függvény a rajzelemtípus szempontjából érvénytelen adatot kap, akkor a rajzelemmel együtt az egész összetett rajzelemet elveti. Egy blokkdefiníciót nem lehet egy másikba ágyazni, és nem hivatkozhat magára sem. Azonban tartalmazhat hivatkozásokat más blokkdefiníciókra. Egy összetett rajzelem összes rajzeleme létezhet a modelltéren és a papírtéren is, de a kettőn egyszerre nem.

Egy 66-os csoportkód csak insert objektumok számára engedélyezett (*attribútumok következnek* jelentéssel). Vonallánc-rajzelemek esetén a 66-os csoportkód értéke 1 (*csomópontok következnek* jelentéssel), és az összes többi rajzelem esetén az alapértelmezett 0 értéket veszi fel. Egy vonallánc-rajzelemet csak egy csomópont-rajzelem követhet.

Az összetett rajzelem egyetlen része sem jelenik meg a rajzon, míg annak definíciója teljessé nem válik. Egy összetett rajzelem létrehozását az **entmake** argumentum nélküli meghívásával szakíthatja meg. Ez törli az ideiglenes fájlt, és `nil` értékkel tér vissza.

A block és endblk rajzelemeket egy új blokkdefiníció létrehozására lehet használni. Az újonnan létrehozott blokkok automatikusan bekerülnek a szimbólumtáblába, ahol aztán lehet rájuk hivatkozni.

Az alkalmazások tetszőlegesen nagy oldalszámú sokszögeket tartalmazó soklaphálókat ábrázolhatnak. Az AutoCAD elemstruktúrája azonban határt szab egy adott lap által meghatározható csomópontok számának. Az ennél összetettebb sokszögeket úgy ábrázolhatja, hogy háromszögekre bontja őket. Az AutoCAD a háromszög alakú lapokat olyan négy csomópontú felületként ábrázolja, amelynél két szomszédos csomópont értéke megegyezik. Annak érdekében, hogy a segédelemek ne zavarják az összkép megjelenését, a háromszögek élei láthatatlanná tehetők. A SOKLAP parancs automatikusan elvégzi ezt a felosztást, míg az alkalmazások által közvetlenül generált soklaphálók esetében az alkalmazásoknak maguknak kell ezt megtenniük.

A segédelemek kialakításának kulcsparamétere a laponkénti csomópontszám. Az alkalmazások számára a PFACEVMAX rendszerváltozó adja meg a laponként felhasználható csomópontok számát. Ennek értéke csak olvasható, és 4-re van beállítva.

Figyelmeztetés Amikor az **entmake** létrehoz egy blokkot, akkor felülírhat egy már létezőt. Az **entmake** függvény nem ellenőrzi a blokkdefiníciós táblát névazonosság esetére, ezért hívása előtt használja a **tblsearch** függvényt annak megállapítására, hogy az új blokknév egyedi-e. Az **entmake** használata azonban hasznos lehet a következőkben leírt névtelen blokkok átdefiniálásában.

Névtelen blokkok

Egy rajz blokkdefiníciós táblája tartalmazhat névtelen blokkokat. A névtelen blokkokat a sraffozási minta és az asszociációs méretezés támogatására hozták létre. Az **entmake** létrehozhatja őket az alkalmazás saját céljaira is, rendszerint olyan rajzelemek tárolására, amelyeket a felhasználó nem tud közvetlenül elérni.

Egy névtelen blokk neve (2-es csoport) *U*nnn* alakú, ahol *nnn* egy AutoCAD által generált szám. Egy névtelen blokk *Blokktípus flag* (70-es csoport) alsó bit értéke 1. Amikor az **entmake** olyan blokkot hoz létre, melynek neve *-gal kezdődik és melynek névtelen bit értéke be van állítva, akkor az AutoCAD névtelen blokként kezeli és hozzárendel egy nevet. A * jelet követő karaktereket az **entmake** figyelmen kívül hagyja. A blokk létrehozása után az **entmake** annak nevével tér vissza. Amennyiben Ön többszörös **entmake** hívással hozzza létre a blokkot, az alábbi sikeres hívás után tér vissza a névvel a függvény.

```
(entmake "endblk")
```

Amikor egy rajzot megnyit, akkor az összes névtelen blokk, amelyre nincs hivatkozás eltűnik a blokkdefiníciós táblából. A hivatkozott (beillesztett) névtelen blokkok nem tűnnek el. Az **entmake** függvény használatával blokkhivatkozást (Insert) hozhat létre egy névtelen

blokkra (a BEILL parancsnak nem adhat át névtelen blokkot). Használhatja az **entmake** függvényt a blokk átdefiniálására is. A blokkon belüli rajzelemeket (de nem magát a blokk rajzelemet) módosíthatja az **entmod** függvénnyel.

Megjegyzés Bár egy hivatkozott névtelen blokk állandóvá válik, nevének számokból álló része változhat a szerkesztési alkalmak között. Az alkalmazások nem hagyatkozhatnak arra, hogy a névtelen blokkok nevei állandóak maradnak.

entmod

Megváltoztatja egy objektum (rajzelem) definíciós adatait

(entmod *elist*)

Az **entmod** függvénynek megadott lista (*elist*) az **entget** által visszaadottal megegyező formátumú és az *elist* argumentumban –1-es csoport által megadott rajzelem adatbázisára vonatkozó információt frissíti fel. Az AutoLISP elsődleges aktualizáló mechanizmusa úgy működik, hogy az **entget** függvény lekérdezi a rajzelem adatait, egy rajzelem definiálásával módosítja a listát és az **entmod** függvénnyel frissíti az adatbázisban levő rajzelemet. Az **entmod** függvény egyaránt módosíthat grafikus és nem grafikus objektumokat is.

```
(setq en (entnext))          Az en-be tölti a rajz első rajzelemét
(setq ed (entget en))        Az ed-be tölti az en rajzelem adatait
(setq ed
  (subst (cons 8 "0")
   (assoc 8 ed)              A fóliacsoportot 0-ra változtatja az ed-ben
    ed
  )
)
(entmod ed)                  Az en rajzelem fóliáját módosítja a rajzon
```

Az **entmod** által végzett módosításoknak vannak bizonyos korlátai. Először is a rajzelem típusa és sorszáma nem módosítható. Amennyiben ezt óhajtja, használja az **entdel** függvényt, és hozzon létre egy új rajzelemet a **command** vagy az **entmake** függvénnyel. Az AutoCAD programnak ismernie kell az elemlista összes hivatkozott objektumát mielőtt az **entmod** függvényt meghívja. Így egy rajzon definiálni kell a szövegstílust, vonaltípust, alak- és blokkneveket, mielőtt egy elemlistában használná őket az **entmod** függvénnyel. Ez alól a fólia nevek kivételek : az **entmod** a szabvány alapértelmezett értékekkel a FÓLIA Új opciójának használatával létrehoz egy új fóliát, ha egy nem definiált fólianévvel találkozik az elemlistában.

A valós értékű rajzelem-mezők esetében (például vastagság) az **entmod** elfogad egész értékeket és azokat lebegőpontos értékké konvertálja. Hasonlóképpen, ha Ön valós számot ad meg egy egész szám típusú

rajzelem-mezőnek (például színszám), akkor az **entmod** csonkolja, és egész számmá konvertálja azt.

Az **entmod** függvény ellenőrzi, hogy a megadott lista összefüggő-e. Amennyiben súlyos hibát talál, az adatbázist nem frissíti, és `nil` értékkel tér vissza. Máskülönben az **entmod** az argumentumaként megadott listával tér vissza. Az **entmod** nem tudja a belső mezők értékét (mint például a rajzelem nevét egy seqend rajzelem –2-es csoportjában) megváltoztatni—az erre irányuló kísérletet figyelmen kívül hagyja.

Amikor az **entmod** felfrissít egy fő rajzelemet, módosítja és felfrissíti annak ábráját a képernyőn (az alárendelt rajzelemeivel együtt). Amikor az **entmod** egy alárendelt rajzelemet frissít (vonallánc-csomópontot vagy blokkattribútumot), akkor az adatbázisban felfrissíti azt, de a képernyőn levő ábra nem változik. Miután egy adott rajzelem alárendelt rajzelemein az összes módosítást elvégezte, az **entupd** függvényt lehet a képernyőn levő ábra frissítésére használni.

Megjegyzés Az **entmod** függvényt nem használhatja egy nézetablak-rajzelem módosítására. Egy rajzelem térbeli láthatóság mezőjének bitjét állíthatja 0 vagy 1 értékre (a nézetablak objektum kivételével). Amennyiben Ön az **entmod** függvényt használja egy blokkon belüli rajzelem módosítására, akkor a módosítás a rajzon látható összes érintett blokkra hatással van.

A csomópontokon végrehajtott **entmod** művelet előtt Önnek ki kell olvasnia vagy át kell írnia a vonallánc-rajzelem fejlécét. Amennyiben az utoljára feldolgozott vonallánc-rajzelem különbözik attól, amelyikhez a csomópont tartozik, akkor a szélességre vonatkozó információ (a 40-es és 41-es csoport) elveszhet.

Figyelmeztetés Az **entmod** függvénnyel blokkokon belüli rajzelemeket is módosíthat, így azonban egy önmagára hivatkozó blokk jöhet létre, amely leállítja az AutoCAD programot.

entnext

A rajzon található következő objektum (rajzelem) nevével tér vissza

(entnext *[ename]***)**

Ha az **entnext** függvényt argumentum nélkül hívja meg, akkor az az adatbázisban található első nem törölt rajzelem nevével tér vissza. Amennyiben az **entnext** függvényt az *ename* argumentummal hívja meg, akkor az *ename* utáni első nem törölt rajzelem nevével tér vissza. Ha nincs következő rajzelem az adatbázisban, akkor `nil` értékkel tér vissza. Az **entnext** függvény egyaránt visszaadja a fő és alárendelt rajzelemeket.

Az **ssget** függvénnyel kiválasztott rajzelemek fő rajzelemek, nem blokkok attribútumai és nem vonalláncok csomópontjai. Az ilyen összetett rajzelemek belső szerkezeteit az alárendelt rajzelemeken keresztüllépkedve az **entnext** függvénnyel érheti el. Amennyiben egyszer hozzájutott egy alárendelt rajzelemhez, akkor ugyanolyan műveleteket végezhet rajta, mint bármely más rajzelemen. Ha az **entnext** függvény segítségével hozzájut egy alárendelt rajzelemhez, megtalálhatja a szülő rajzelemet úgy, hogy az **entnext** függvénnyel addig lépked előre, amíg egy seqend rajzelemet nem talál. A –2-es csoport kivonatával hozzájuthat a fő rajzelem nevéhez.

`(setq e1 (entnext))`	*Az* `e1`*-be tölti a rajzon található első rajzelem nevét*
`(setq e2 (entnext e1))`	*Az* `e2`*-be tölti az* `e1` *rajzelemet követő rajzelemet*

entsel

Felkínálja a felhasználónak, hogy egy pont megadásával válasszon egy objektumot (rajzelemet)

(entsel ***[prompt]*****)**

Az **entsel** függvény egy olyan listával tér vissza, amelynek első eleme a kiválasztott objektum neve, a második eleme pedig annak a pontnak a koordinátája (FKR-ben kifejezve), amellyel kiválasztotta az objektumot. Amennyiben egy karakterláncot is megad *prompt* argumentumként, akkor az kéri a felhasználót az objektum kiválasztására. Máskülönben a Válasszon objektumot lesz az alapértelmezett prompt .

Az **entsel** által visszaadott listát meg lehet adni az AutoCAD bármelyik objektumválasztó promptjára válaszolva. Az AutoCAD ezt úgy kezeli, mintha a kijelölt objektumot az adott pontra való mutatással választotta volna ki.

A következő AutoCAD parancssorozat az **entsel** függvény használatát és az általa visszaadott listát illusztrálja:

Parancs: **vonal**
Ponttól...: **1,1**
...Pontig: **6,6**
...Pontig: ↵
Parancs: **(setq e (entsel "Kérem válasszon egy objektumot: "))**
Kérem válasszon egy objektumot: **3,3**
(<Elem neve: 60000014> (3.0 3.0 0.0))

Az objektumokon végzett műveleteknél néha előfordul, hogy ki akar választani egy objektumot, és egyidejűleg meg akarja adni azt a pontot, amellyel az objektumot kiválasztotta. Erre találhat példákat az

AutoCAD programban a tárgyraszternél és MEGTÖR, METSZ és az ELÉR parancsoknál. Az **entsel** függvény lehetővé teszi az AutoLISP programok számára ilyen művelet végrehajtását. Egy objektumot egy pont megadásával választ ki. A függvény figyelmen kívül hagyja az aktuális Traszter beállításokat (nincs objektum raszter), hacsak azt Ön kifejezetten nem kéri, amikor a függvényt használja. Az **entsel** függvény tiszteletben tartja az előzetes **initget** hívásból származó kulcsszavakat.

entupd

Egy objektum (rajzelem) ábráját frissíti fel a képernyőn

(entupd *ename*)

Amikor egy vonallánc-csomópontot vagy blokkattribútumot módosít az **entmod** függvénnyel, akkor a függvény az összetett rajzelemet nem frissíti fel a képernyőn. Az **entupd** függvényt használhatja egy módosított vonallánc vagy blokk frissítésére a képernyőn. Ezt a függvényt meghívhatja egy vonallánc vagy blokk bármelyik részének rajzelem nevével; nem kell, hogy az fő rajzelem legyen. Bár az **entupd** függvényt vonalláncok és attribútumokkal rendelkező blokkok számára hozták létre, bármilyen rajzelemmel meg lehet hívni. A függvény mindig újra generálja a rajzelemet a képernyőn, az összes alárendelt rajzelemmel együtt.

Megjegyzés Amennyiben az **entupd** függvényt egy másikba ágyazott rajzelemmel (egy blokkon belüli rajzelemmel) használja; vagy egy olyan blokkal, amely egymásba ágyazott rajzelemeket tartalmaz, akkor előfordulhat, hogy néhány rajzelem nem generálódik újra. A teljes újra generálás biztosítása érdekében használja a REGEN parancsot.

Feltéve, hogy a rajz első rajzeleme egy vonallánc több csomóponttal, akkor

```
(setq e1 (entnext))          Az e1-be tölti a vonallánc rajzelem nevét
(setq e2 (entnext e1))       Az e2-be tölti annak első csomópontját
(setq ed (entget e2))        Az ed-be tölti a csomópontok adatait
(setq ed
  (subst '(10 1.0 2.0)
    (assoc 10 ed)            (1,2)-re változtatja a csomópont helyét
    ed                       az ed-ben
  )
)
(entmod ed)                  Elmozdítja a csomópontot a rajzon
(entupd e1)                  Újra generálja az e1 vonalláncot
```

eq

Két kifejezés azonosságát állapítja meg

(eq *expr1 expr2*)

Az **eq** függvény eldönti, hogy az *expr1* és *expr2* kifejezéseket ugyanahhoz az objektumhoz rendelték-e (például a **setq** függvény segítségével). Amennyiben a két kifejezés azonos, akkor T, máskülönben nil a visszatérési érték.

Két lista azonosságát állapíthatja meg ezzel a függvénnyel. Például az alábbi értékadások esetén

```
(setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
```

akkor

(eq f1 f3) *eredménye* nil, f1 *és* f3 *nem ugyanaz a lista*
(eq f3 f2) *eredménye* T, f3 *és* f2 *pontosan ugyanaz a lista*

További tudnivalók

= (egyenlő) és **equal** függvény

equal

Két kifejezés egyenlőségét állapítja meg

(equal *expr1 expr2 [fuzz]*)

Az **equal** függvény eldönti, hogy az *expr1* és *expr2* kifejezések értéke egyenlő-e. Mikor két valós számot (vagy két valós számból álló listát, mint pontokat) hasonlít össze, a két *azonos* szám kis mértékben eltérhet, ha különböző módszerekkel számolta ki őket. Ezért használhat egy opcionális szám argumetumot, *fuzz*-t, amellyel megadja azt a legnagyobb értéket, amellyel *expr1* és *expr2* kifejezések eltérhetnek anélkül, hogy a függvény különbözőnek venné őket .

Például az alábbi értékadások esetén

```
(setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
(setq a 1.123456)
(setq b 1.123457)
```

akkor

```
(equal f1 f3)          eredménye T
(equal f3 f2)          eredménye T
(equal a b)            eredménye nil
(equal a b 0.000001)   eredménye T
```

Noha az **equal** függvény által egyenlőnek talált két listát az **eq** függvény nem mindig találja azonosnak, az **equal** által egyenlőnek ítélt atomokat az **eq** függvény is mindig azonosnak veszi. Ha azonban az **eq** függvény listákat vagy atomokat azonosnak ítél, akkor az **equal** függvény is egyenlőnek találja őket.

További tudnivalók

= (egyenlő) és **eq** függvény

error

Felhasználó által definiálható hibakezelő függvény

(*error* *string*)

Amennyiben ***error*** értéke nem `nil`, akkor függvényként hajtódik végre, valahányszor az AutoLISP hiba feltétel létezik. Az AutoCAD egy argumetumot ad át ***error*** függvénynek, amely egy karakterlánc, és a hiba leírását tartalmazza.

A következő függvény ugyanazt teszi, mint az AutoLISP szabvány hibakezelője. Megadja a hibát és a leírását.

```
(defun *error* (üzenet)
  (princ "hiba: ")
  (princ üzenet)
  (princ)
)
```

Az Ön ***error*** függvénye tartalmazhat argumentum (például `(command)`) nélküli **command** függvényt . Ez elveti a korábban **command** függvénnyel meghívott AutoCAD parancsot.

További tudnivalók

"Hibakezelés" című résznél, a 216. oldalon egy példát láthat a hibakezelésre, amely az **exit** és az **quit** függvények által visszaadott karakterláncot teszteli.

eval

Egy AutoLISP kifejezés kiértékelt eredményével tér vissza

(eval *expr*)

Például az alábbi értékadás esetén

```
(setq a 123)
(setq b 'a)
```

akkor

```
(eval 4.0)          eredménye 4.0
(eval (abs -10))    eredménye 10
(eval a)            eredménye 123
(eval b)            eredménye 123
```

exit

Megszakítja az aktuális alkalmazást

(exit)

Amennyiben **exit** függvényt meghívják, akkor az a quit/exit megszakítás hibaüzenetet jeleníti meg és visszatér az AutoCAD Parancsprompthoz.

További tudnivalók

quit függvény

exp

A konstans *e* (egy valós szám) adott hatványával tér vissza (a természetes alapú logaritmus ellentetje)

(exp *num*)

```
(exp 1.0)           eredménye 2.71828
(exp 2.2)           eredménye 9.02501
(exp -0.4)          eredménye 0.67032
```

expand

Egy adott számú szegmens által kért csomóponti (node) helyet jelöl ki

(expand *int*)

További tudnivalók

További információ az **expand** függvényről "Manuális allokáció" című résznél, az 566. oldalon

expt

Egy szám adott hatványával tér vissza

(expt *munber power*)

Amennyiben mindkét argumentum egész, akkor az eredmény egy egész szám. Máskülönben az eredmény egy valós szám. Például:

```
(expt 2 4)          eredménye 16
(expt 3.0 2.0)      eredménye 9.0
```

fill_image

Egy kitöltött téglalapot rajzol az éppen aktív párbeszédpanel ábrazónájába

(fill_image *x1 y1 wid hgt color*)

A **fill_image** függvényt a **start_image** és **end_image** függvények között kell meghívni. A *color* paraméter egy AutoCAD színszám, vagy egy az alábbi táblázatban bemutatott logikai színszám.

A szín attribútum szimbólikus nevei

Színszám	*ADI mnemonik*	*Leírás*
–2	BGLCOLOR	Az aktuális háttér az AutoCAD grafikus képernyőjén
–15	DBGLCOLOR	A aktuális párbeszédpanel hátterének színe
–16	DFGLCOLOR	Az aktuális párbeszédpanel előtérének színe (szöveg)
–18	LINELCOLOR	Az aktuális párbeszédpanel vonalának színe

A négyszög első (bal felső) sarka az (*x1*,*y1*) helyen és a második (jobb alsó) sarka az elsőtől (*wid*,*hgt*) relatív távolságra van (a *wid* és a *hgt* argumentumok értéke pozitív kell, hogy legyen). Az origó (0,0) pont az ábra bal felső sarka. A jobb alsó sarok koordinátáit a **dimx_tile** és **dimy_tile** méretfüggvények hívásával kaphatja meg.

findfile

Megkeresi az adott fájlt az AutoCAD könyvtár útvonalban

(findfile *filename*)

A **findfile** függvény nem tételez fel semmit a *filename* típusát illetve kiterjesztését illetően. Amennyiben a *filename* nem adja meg a meghajtó/könyvtár előtagot, akkor a **findfile** az AutoCAD könyvtár útvonalában keresi a fájlt. Amennyiben megadja a meghajtó/könyvtár előtagot, akkor a **findfile** csak azt a könyvtárat nézi meg. A **findfile** függvény mindig a teljes meghajtó/könyvtár/fájl nevet adja vissza, vagy `nil` értéket, ha nem találja az adott fájlt.

Tekintse meg az alábbi példákat. Ha az aktuális könyvtár az */acad* és tartalmazza az *abc.lsp* nevű fájlt, akkor a.

`(findfile "abc.lsp")` *eredménye* `"/acad/abc.lsp"`

Ha Ön egy rajzot editál az */acad/drawings* alkönyvtárban, az ACAD környezeti változó értéke */acad/support*, és az *xyz.txt* fájl csak az */acad/support* alkönyvtárban létezik.

`(findfile "xyz.txt")` *eredménye* `"/acad/support/xyz.txt"`

Ha a *nincs* fájl nem található a könyvtárkeresési útvonal egyik alkönyvtárában sem, akkor a

`(findfile "nincs")` *eredménye* `nil`

A **findfile** által visszaadott teljes nevet az **open** függvény tudja használni.

fix

Egy valós számot a hozzá legközelebbi, kisebb egésszé konvertál

(fix *num*)

A **fix** függvény a *number* argumentummal megadott számot törtrészének elhagyásával a hozzá legközelebbi egész számra csonkolja.

```
(fix 3)                    eredménye  3
(fix 3.7)                  eredménye  3
```

Megjegyzés Ha a *number* nagyobb, mint lehető legnagyobb egész (+2 147 483 647 vagy –2 147 483 648 egy 32 bites platformon), akkor a **fix** egy csonkolt törtszámot ad vissza (bár AutoLISP és AutoCAD között közvetített egész számok 16 bites értékre korlátozódnak).

float

Egy szám valós számmá konvertált értékével tér vissza

(float *num*)

```
(float 3)                  eredménye  3.0
(float 3.75)               eredménye  3.75
```

foreach

Egy lista minden elemére kiértékeli a kifejezéseket

(foreach *name list expr...*)

A *list* argumentumon áthaladva, minden elemet hozzárendel a *name* argumentumhoz, és a lista minden elemére kiértékeli az *expr* kifejezést. Tetszőleges számú *expr* kifejezést meg lehet adni. A **foreach** függvény az utoljára kiértékelt *expr* értékével tér vissza.

```
(foreach n '(a b c) (print n))
```

egyenlő

```
(print a)
(print b)
(print c)    és eredménye  c
```

azzal az eltéréssel, hogy a **foreach** csak az utoljára kiértékelt kifejezés értékével tér vissza.

gc

Hulladékgyűjtő folyamatot indít, amely felszabadítja a nem használt csomópontokat

```
(gc)
```

További tudnivalók

"Csomóponti terület" című résznél, a 564. oldalon a hulladékgyűjtő folyamat részletesebb magyarázata

gcd

Visszaadja két egész szám legnagyobb közös osztóját

```
(gcd int1 int2)
```

Az *int1* és *int2* argumentumnak 0-nál nagyobb egésznek kell lenni.

```
(gcd 81 57)          eredménye 3
(gcd 12 20)          eredménye 4
```

get_attr

Visszakeresi egy párbeszédpanel-attribútum DCL fájlban lévő értékét

```
(get_attr key attribute)
```

A *key* argumentum egy olyan, a zónát megadó karakterlánc, amely megkülönbözteti a kis- és nagybetűket . Az *attribute* argumentum úgy adja meg az attribútum nevét, ahogyan az a zóna DCL leírásában szerepel. A *key* és *attribute* argumentumok is karakterláncok. A visszaadott érték az attribútum kezdeti értéke, amit annak DCL leírása határoz meg; ezt egyáltalán *nem* befolyásolja a zóna állapotának felhasználói bevitel vagy **set_tile** hívás útján történt megváltoztatása. Az attribútum értékével mint karakterlánccal tér vissza.

get_tile

Visszakeresi a zóna pillanatnyi értékét

(get_tile *key*)

A *key* argumentum egy olyan, a zónát megadó karakterlánc, amely a különbséget tesz a kis- és nagybetűk között. A zóna értékét karakterláncként adja vissza.

getangle

Vár, amíg a felhasználó megad egy szöget, majd a szög radiánban kifejezett értékével tér vissza

(getangle *[pt] [prompt]*)

A *pt* argumentum egy kétdimenziós bázispont az aktuális FKR-ben, a *prompt* pedig egy karakterlánc, amely promptként jelenik meg a képernyőn. Ha a *pt* meg van adva, akkor ez lesz a két pont közül az első, azért, hogy a felhasználó egy másik pontra mutatással meg tudja adni a szöget az AutoLISP rendszernek . Megadhat egy háromdimenziós bázispontot is, de a függvény mindig az aktuális szerkesztősíkon méri a szöget.

A **getangle** függvény a szöget nulla radián iránnyal (ezt az ANGBASE rendszerváltozó állítja be) méri az óramutató járásával ellentétes irányban növekvő értékkel. Az eredményül kapott szög radiánokban van kifejezve, és az aktuális szerkesztősíkban (az aktuális FKR *XY* síkja az aktuális kiemelési magasságban) értendő.

A felhasználó a szöget az AutoCAD aktuális szögmértékegységében kifejezve, bebillentyűzéssel is megadhatja. Jóllehet a szögek aktuális kifejezési formátuma fok, grad vagy bármely más mértékegység lehet, a függvény mindig radiánban kifejezett szögértékkel tér vissza. A felhasználó meg is mutathatja a szöget az AutoLISP rendszernek, úgy, hogy a grafikus képernyőn kijelöl két kétdimenziós pontot. Az AutoCAD az első pontból kiindulva egy "gumivonalat" húz a szálkereszt aktuális pozíciójáig, láthatóvá téve ezzel a szöget.

Fontos, hogy megértse a különbséget a **getangle** függvény bemeneti szöge és visszatérő szögértéke között. A **getangle** függvénynek megadott szögek értéke az ANGDIR és ANGBASE rendszerváltozók aktuális beállításán alapszik. Az így megadott szöget azután a függvény az ANGDIR figyelembevétele nélkül, az óramutató járásával ellentétes irányban méri, az ANGBASE aktuális beállítása szerinti nulla radiánhoz.

```
(setq ang (getangle))
(setq ang (getangle '(1.0 3.5)))
(setq ang (getangle "Melyik irányban? "))
(setq ang (getangle '(1.0 3.5) "Melyik irányban? "))
```

A felhasználó a **getangle** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

További tudnivalók

Illusztráció és a **getorient** függvénnyel való összehasonlítás

getcfg

Az *acad.cfg* fájl AppData szakaszának alkalmazás adataival tér vissza

(getcfg *cfgname*)

A *cfgname* argumentum egy karakterlánc, amely megnevezi a visszaadandó szakasz- és paraméterértéket. Amennyiben a *cfgname* nem érvényes, a **getcfg** `nil` értékkel tér vissza. A *cfgname* argumentumnak a következő formátumú karakterláncnak kell lennie :

`"AppData/alkalmazásnév/szakasznév/.../param_név"`

Például feltéve, hogy a WallThk paraméter értéke az AppData/ArchStuff szakaszban 8,

`(getcfg "AppData/ArchStuff/WallThk")` *eredménye* `"8"`

További tudnivalók

setcfg függvény

getcorner

Vár, amíg a felhasználó megadja egy négyszög második sarkát

(getcorner *pt [prompt]*)

A **getcorner** függvény argumentumként az aktuális FKR egy *pt* bázispontjának megadását várja, és ebből a pontból a szálkereszt mozgatásával egy téglalapot rajzol a képernyőre. A *prompt* argumentum egy karakterlánc, amely promptként fog megjelenni a képernyőn. A getcorner függvény, a **getpoint** függvényhez hasonlóan, az aktuális FKR egy pontjával tér vissza. Ha a felhasználó egy háromdimenziós pontot ad meg, akkor annak *Z* koordinátáját a függvény figyelmen kívül hagyja, és az aktuális kiemelési síkot fogja *Z* koordinátaként használni.

A felhasználó a **getcorner** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

getdist

Vár, amíg a felhasználó megad egy távolságot

(getdist *[pt] [prompt]*)

A *pt* argumentum az aktuális FKR egy két- vagy háromdimenziós bázispontja. Ha a *pt* meg van adva, akkor a két pont közül az lesz az első, és a függvény csak a második pont megadását kéri. A *prompt* argumentum egy karakterlánc, amely promptként fog megjelenni.

Ha a bázispont adott, a felhasználó egy, különben pedig két pont kiválasztásával adhatja meg a távolságot. A felhasználó be is billentyűzheti a távolságot az AutoCAD aktuális mértékegységének formátumában. Jóllehet a távolságok aktuális kifejezési formátuma láb és hüvelyk (építészet) is lehet, a **getdist** függvény mindig valós számként adja vissza a távolságot.

A **getdist** függvény gumivonalat húz az első ponttól a szálkereszt aktuális pozíciójáig, hogy szemléletessé tegye a távolságot.

Háromdimenziós pont megadása esetén a visszatérési érték egy térbeli (3D) távolság. Az **initget** függvény 64 értékű bitjének beállításával a **getdist** utasítható a háromdimenziós pontok *Z* koordinátájának figyelmen kívül hagyására, ekkor a visszatérési érték egy síkbeli (2D) távolság lesz.

```
(setq dist (getdist))
(setq dist (getdist '(1.0 3.5)))
(setq dist (getdist "Mekkora a távolság "))
(setq dist (getdist '(1.0 3.5) "Mekkora a távolság? "))
```

A felhasználó a **getdist** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

getenv

Egy környezeti változóhoz rendelt karakterlánc értékével tér vissza

(getenv *variable-name*)

A *variable-name* argumentum az olvasni kívánt változó nevét megadó karakterláncot tartalmazza. Amennyiben ez a változónév nem létezik, a **getenv** függvény `nil` értékkel tér vissza.

Például, ha az ACAD környezeti változó értéke */acad/support*, a NINCS környezeti változó pedig nem létezik, akkor

```
(getenv "ACAD")     eredménye "/acad/support"
(getenv "NINCS")    eredménye nil
```

Megjegyzés UNIX operációs rendszerek esetében az ACAD és az `acad` két különböző környezeti változóra való hivatkozást jelent, mivel ezek az operációs rendszerek különbséget tesznek a kis- és nagybetűk között.

getfiled

Az AutoCAD fájlkiválasztásra szolgáló párbeszédpaneljében egy fájlnév megadását kéri a felhasználótól, és a fájl nevét adja vissza

(getfiled *title default ext flags*)

A *title* argumentum a párbeszédpanel címét, a *default* az alapértelmezés szerinti fájlnevet (amely üres karakterlánc [""] is lehet), az *ext* pedig a fájlnév alapértelmezés szerinti kiterjesztését adja meg. Amennyiben az *ext* egy üres karakterlánc [""], akkor a kiterjesztés alapértelmezés szerint * (összes fájltípus) lesz. Amennyiben a `dwg` fájltípus van megadva az *ext* argumentumban, akkor a függvény az ábrákat tartalmazó fájlok listáját jeleníti meg a párbeszédpanelben. A *flags* argumentum egy egész típusú érték (bit-kódolású mező), amely a párbeszédpanel viselkedését szabályozza. Amennyiben egyidejűleg több feltételt kíván kikötni, akkor adja össze az értékeket; így egy 0 és 15 közé eső *flags* értéket állíthat elő.

Érték = 1 (0. bit)

Ezt a bitet akkor kell beállítani, ha egy *létrehozni* kívánt új fájl nevét kéri. Egy *megnyitni* kívánt, már meglévő fájl nevére irányuló kérés esetén nem szabad beállítani ezt a bitet. Ez utóbbi esetben, ha a felhasználó egy még nem létező fájl nevét billentyűzi be, akkor a párbeszédpanel alján egy hibaüzenet jelenik meg.

Amennyiben ez a bit be van állítva, és a felhasználó egy már létező fájl nevét választja ki, akkor az AutoCAD egy figyelmeztető ablakot jelenít meg, amelyben választási lehetőséget kínál a felhasználónak, hogy folytatja vagy elveti a műveletet.

Érték = 2 (1. bit)

Letiltja a Bebill. nyomógombot. Ez a bit automatikusan beállításra kerül, ha a **`getfiled`** hívására egy másik párbeszédpanel aktív állapotában kerül sor. (Máskülönben a másik párbeszédpanelt is eltüntetné a képernyőről.)

Amennyiben ez a bit nincs beállítva, a Bebill. nyomógomb engedélyezett. Ha a felhasználó azt választja ki, a párbeszédpanel eltűnik, és a **`getfiled`** az 1 értékkel tér vissza.

Érték = 4 (2. bit)

Lehetővé teszi, hogy a felhasználó a fájlnak tetszőleges kiterjesztést adjon meg, vagy egyáltalán ne adjon meg kiterjesztést.

Amennyiben ez a bit nincs beállítva, akkor a **getfiled** *csak* az **ext** argumentumban megadott kiterjesztést fogadja el, illetve azt fűzi hozzá a fájl nevéhez, ha a felhasználó nem ad meg kiterjesztést a Fájl szerkesztőablakban.

Érték = 8 (3. bit)

Amennyiben ez a bit be van állítva, a 0. bit pedig *nincs,* a **getfiled** keresni kezdi a megadott fájlnevet a könyvtárakban. Amennyiben a fájlt, illetve az azt tartalmazó könyvtárat a könyvtárkeresési útvonalon találja meg, akkor az elérési útvonalat leválasztja, és csak a fájlnevet adja vissza. (Nem választja le az útvonalnevet, ha talál ugyanilyen nevű fájlt, de egy másik könyvtárban.)

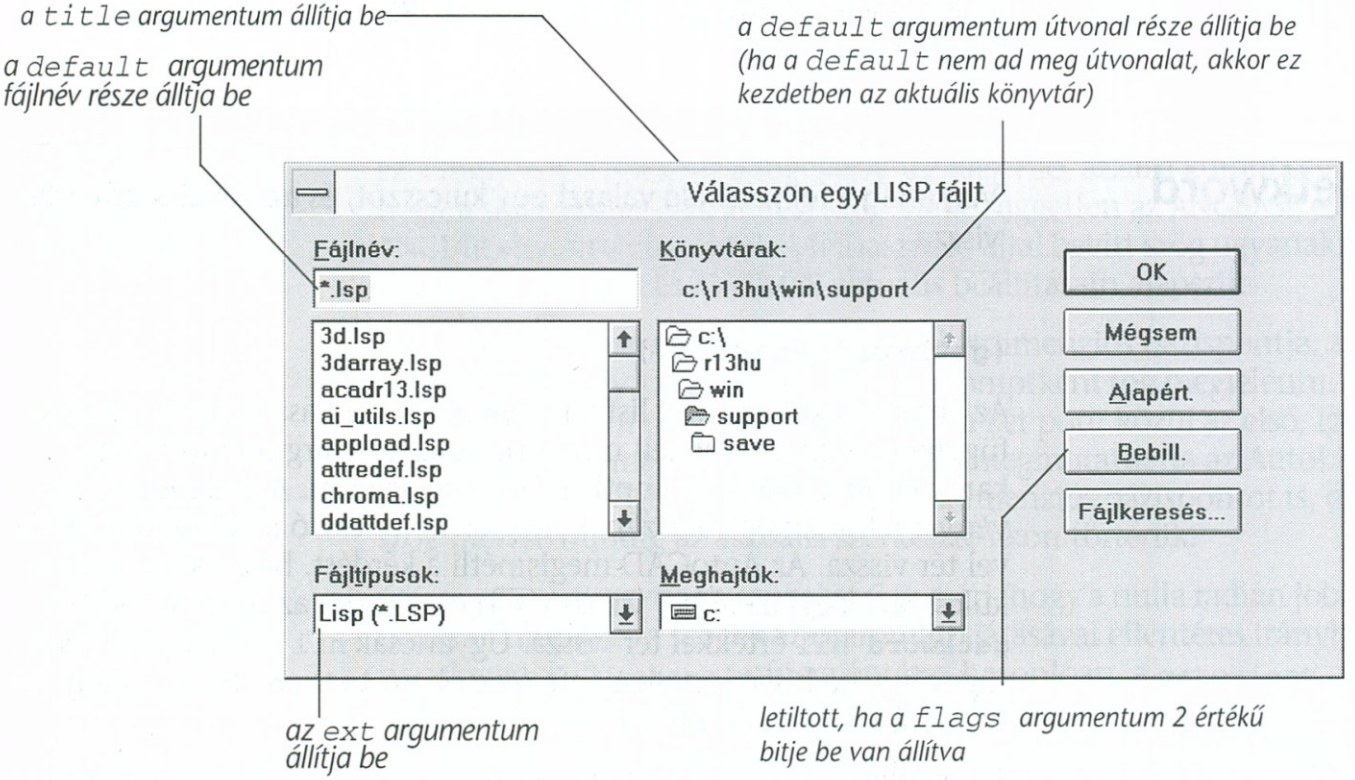
Amennyiben ez a bit nincs beállítva, akkor a **getfiled** függvény az elérési útvonalat is magában foglaló teljes fájlnévvel tér vissza.

Akkor állítsa be ezt a bitet, ha egy olyan meglévő fájl megnyitására használja a párbeszédpanelben, amelynek nevét el kívánja menteni a rajzban (vagy egyéb adatbázisban).

Amennyiben a felhasználó megad egy fájlnevet a párbeszédpanelben, akkor a **getfiled** függvény a fájlnevet megadó karakterláncot adja vissza; máskülönben `nil` értékkel tér vissza.

A következő **getfiled** hívás hatására az alábbi Válasszon egy Lisp fájlt című párbeszédpanel jelenik meg:

```
(getfiled "Válasszon egy Lisp fájlt" "/acadr13/win/support/" "lsp"
8)
```



Minta getfile párbeszédpanel

A **getfiled** függvény egy párbeszédpanelt jelenít meg, amely az elérhető és a megadott kiterjesztéssel rendelkező fájlok felsorolását tartalmazza. Ennek segítségével különböző meghajtókat és könyvtárakat "átfésülve" kiválaszthat egy meglévő fájlt, vagy megadhatja egy új fájl nevét.

getint

Vár, amíg a felhasználó egy egész értéket ad meg, és ezzel az egésszel tér vissza

(getint *[prompt]*)

A *prompt* argumentum egy opcionális karakterlánc, amely promptként fog megjelenni. A **getint** függvény az egész értékkel vagy `nil` értékkel tér vissza.

```
(setq num (getint))
(setq num (getint "Adjon meg egy számot: "))
```

A **getint** függvénynek átadott értékek a –32768 és +32767 közti tartományba eshetnek. A felhasználó a **getint** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

További tudnivalók

"A getxxx függvények" című résznél, a 233. oldalon és **initget** függvény

getkword

Vár, amíg a felhasználó választ egy kulcsszót, és azt a kulcsszót adja vissza

(getkword ***[prompt]*****)**

Az érvényes kulcsszavak listáját a **getkword** hívása előtt az **initget** függvény segítségével kell megadni. A *prompt* argumentum egy karakterlánc, amely promptként fog megjelenni. A **getkword** függvény a felhasználó válaszának megfelelő kulcsszó karakterlánc értékével tér vissza. Az AutoCAD megismétli a kérdést, ha a válasz nem a kulcsszavak egyike. Üres válasz (↵) esetén (ha az megengedett) a **getkword** `nil` értékkel tér vissza. Ugyancsak `nil` értékkel tér vissza a függvény, ha hívását nem előzte meg olyan **initget** hívás, amely egy vagy több kulcsszót állított be.

A következő példában az **initget** függvény olyan hívása szerepel, amely kulcsszavak (Igen és Nem) listáját állítja fel, valamint letiltja az üres választ (*bits* értéke 1) a soron következő **getkword** függvény számára:

```
(initget 1 "Igen Nem")
(setq x (getkword "Biztos? (Igen vagy Nem) "))
```

A fenti programrészlet megkérdezi a felhasználót, és válaszától függően az `x` szimbólumot `Igen`-re vagy `Nem`-re állítja. Amennyiben a válasz egyik kulcsszónak sem felel meg, vagy a felhasználó üres választ ad, akkor az AutoCAD a *prompt* argumentumban megadott karakterláncot megjelenítve megismétli a kérdést. Amennyiben a *prompt* argumentum nincs megadva, az AutoCAD ezt a promptot jeleníti meg:

Próbálja meg újból:

A felhasználó a **getkword** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

További tudnivalók

"A getxxx függvények" című résznél, a 233. oldalon és **initget** függvény

getorient

Vár, amíg a felhasználó megad egy szöget, és azt a szöget adja vissza radiánban

(getorient *[pt] [prompt]***)**

Ez a függvény a **getangle** függvényhez hasonló, azzal a különbséggel, hogy a **getorient** visszatérési szögértéke független az ANGBASE és ANGDIR renszerváltozóktól. A felhasználó által bevitt szög ugyanakkor szintén az ANGDIR és ANGBASE aktuális beállításain alapszik.

A *pt* argumentum az aktuális FKR egy kétdimenziós bázispontja, a *prompt* pedig egy karakterlánc, amely promptként fog megjelenni. Ha a *pt* argumentum meg van adva, ez lesz a két pont közül az első, így a felhasználó egy másik pont kijelölésével megmutathatja az AutoLISP rendszernek a szöget. Megadhat háromdimenziós bázispontot is, de a szög mérése mindig az aktuális szerkesztősíkon történik.

A **getorient** függvény a szögeket úgy méri, hogy a nulla radián jobbra (keletre) mutat, és a szögek az óramutató járásával ellentétes irányban növekszenek. A **getangle** függvényhez hasonlóan, a **getorient** a szögek visszatérési értékét radiánban fejezi ki az aktuális szerkesztősíkban értelmezve. A **getorient** függvénynek megadott szögek értéke az ANGDIR és ANGBASE aktuális beállításán alapszik. A megadott szöget az óramutató járásával ellentétes irányban, jobbra mutató nulla radiánnal (az ANGDIR és ANGBASE figyelembe vétele nélkül) méri a függvény. Ezért, ha Ön a MÉRTEGYS paranccsal, vagy az ANGBASE illetve az ANGDIR rendszerváltozókkal más nulla irányt, vagy a szögek növekedésének más irányát állította be, akkor bizonyos átszámításokat kell végeznie.

Használja a **getangle** függvényt, ha elforgatási értékre (relatív szögre) van szüksége. Amennyiben egy iránymeghatározásra (abszolút szögre) van szüksége, akkor a **getorient** függvényt használja.

A felhasználó a **getorient** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

További tudnivalók

"A getxxx függvények" című részнél, a 233. oldalon és **getangle** illetve **initget** függvények

getpoint

Vár, amíg a felhasználó megad egy pontot, és azzal a ponttal tér vissza

(getpoint *[pt] [prompt]***)**

A *pt* argumentum egy két- vagy háromdimenziós bázispont az aktuális FKR-ben, a *prompt* pedig egy karakterlánc, amely promptként fog megjelenni. A felhasználó a pontot kijelölheti a képernyőn, vagy megadhatja azt koordinátáinak az aktuális mértékegység-formátumban történő bebillentyűzésével. Amennyiben a *pt* meg van adva, akkor az AutoCAD egy "gumivonalat" húz ebből a pontból a szálkereszt aktuális pozíciójáig. A függvény visszatérési értéke az aktuális FKR-ben kifejezett háromdimenziós pont lesz.

```
(setq p (getpoint))
(setq p (getpoint "Hol? "))
(setq p (getpoint '(1.5 2.0) "Második pont: "))
```

A felhasználó a **getpoint** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

További tudnivalók

"A getxxx függvények" című résznél, a 233. oldalon és **getcorner** illetve **initget** függvények

getreal

Vár, amíg a felhasználó megad egy valós számot, és azt a valós számot adja vissza

(getreal *[prompt]*)

A *prompt* argumentum egy karakterlánc, amely promptként fog megjelenni.

```
(setq val (getreal))
(setq val (getreal "Méretarány-tényező: "))
```

A felhasználó a **getreal** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

getstring

Vár, amíg a felhasználó megad egy karakterláncot, és azt a karakterláncot adja vissza

(getstring *[cr]* *[prompt]*)

Amennyiben a *cr* argumentum meg van adva, és értéke nem `nil`, a megadott karakterlánc szóközöket is tartalmazhat (és a ↵ billentyűvel kell lezárni). Egyéb esetekben a bevitt karakterlánc egy szóközzel vagy a ↵ billentyűvel zárható le. A prompt argumentum egy karakterlánc, amely promptként fog megjelenni.

Amennyiben a karakterlánc hossza meghaladja a 132 karaktert, a függvény csak az első 132 karakterrel fog visszatérni. Amennyiben a megadott karakterlánc visszahajló törtjelet (\) tartalmaz, ezt a függvény két visszahajló törtjellé (\\) konvertálja. Erre azért van szükség, hogy a visszatérési érték más függvények által használható fájlneveket is tartalmazhasson (elérési útvonallal együtt),

```
(setq s (getstring "Mi az Ön keresztneve? "))
```

a **Pál** válasz eredménye `"Pál"`

```
(setq s (getstring T "Mi az Ön teljes neve? "))
```

a **Kovács Pál** válasz eredménye `"Kovács Pál"`

```
(setq s (getstring "Adja meg a fájl nevét: "))
```

a **\acad\mydwg** válasz eredménye `"\\acad\\mydwg"`

A felhasználó a **getstring** függvény kérésére válaszolva nem írhat be egy másik AutoLISP kifejezést.

További tudnivalók

getkword függvény, ahol egy rutin található, amely arra kéri a felhasználót, hogy több ismert opció közül adjon meg egyet (egy kulcsszót)

getvar

Egy AutoCAD rendszerváltozó értékét adja vissza

(getvar *varname*)

A *varname* argumentum egy karakterlánc, amely megadja a rendszerváltozót. Amennyiben *a varname* érvénytelen rendszerváltozó, akkor a **getvar** `nil` értékkel tér vissza.

Tételezzük fel például, hogy a legutoljára megadott lekerekítő ív sugara 0.25 egység volt,

```
(getvar "FILLETRAD")        eredménye  0.25
```

A jelenleg használható AutoCAD rendszerváltozók felsorolása az *AutoCAD on-line súgójában* található.

További tudnivalók

setvar függvény

graphscr

Az AutoCAD grafikus képernyőjét jeleníti meg

```
(graphscr)
```

A **graphscr** függvény mindig `nil` értékkel tér vissza.

Ez egyenértékű a GRAFKÉP paranccsal, vagy a Képváltó billentyű lenyomásával. A **graphscr** ellentetje a **textscr** függvény.

grclear

Letörli az aktuális nézetablakot

```
(grclear)
```

A **grclear** függvény mindig `nil` értékkel tér vissza.

Ez a függvény érintetlenül hagyja a parancs-/prompt-,állapot- és menüterületeket. A grafikus képernyő korábbi tartalmát a **redraw** függvénnyel állíthatja vissza. Egyképernyős rendszerek esetében a **grclear** függvény a nézetablak törlése előtt először átvált szöveges képernyőről grafikus képernyőre.

grdraw

Egy vektort rajzol két pont közé az aktuális nézetablakban

```
(grdraw from to color [highlight])
```

A *from* és *to* argumentumok két- vagy háromdimenziós pontok (két vagy három valós számból álló listák), amelyek a vektor végpontjait határozzák meg az aktuális FKR-ben. Az AutoCAD a vektor képernyőn kívülre eső részét lehagyja. A **grdraw** függvény a vektort az egész típusú *color* argumentum által megadott színnel rajzolja meg, –1-gyel jelölve az *XOR tintát*, amely az átrajzolt objektumok színét a kiegészítő szinükre fordítja át, saját magát pedig kioltja. Amennyiben az egész típusú *highlight* argumentum meg van adva és értéke nullától különböző, akkor a vektor rajzolása kiemelt megjelenítéssel történik, a megjelenítő eszköz alapértelmezés szerinti kiemelési módszerének megfelelően (általában szaggatott vonallal). A *highlight* elhagyása, vagy nulla értéke esetén a **grdraw** függvény a normál megjelenítési módot alkalmazza. A függvény mindig `nil` értékkel tér vissza.

További tudnivalók

grvecs függvény, ahol egy több vektort rajzoló rutin látható

grread

Értékeket olvas be az AutoCAD bármelyik beviteli eszközéről

(grread *[track] [allkeys [curtype]]*)

Erre a függvényre csak igen speciális AutoLISP rutinokban lehet szükség. Az AutoLISP által igényelt adatbevitel a legtöbb esetben a különböző **getxxx** függvények útján történik.

Amennyiben a *track* argumentum meg van adva és értéke nem nil, akkor a pozícionáló eszköz mozgatása közben is megkaphatja annak aktuális koordinátáit. Amennyiben az *allkeys* argumentum jelen van, a **grread** a megadott kódtól függő funkciót valósít meg. A *curtype* argumentum a képernyőn megjelenő kurzor típusának beállítására szolgál. Az *allkeys* és *curtype* argumentumok egyaránt egész típusúak és az alábbiakban található részletes magyarázatuk:

allkeys Az *allkeys* bit-kódolású értékei összeadhatóak, mely által összetett funkciók hozhatók létre.

1 (0. bit) *Vontatási mód* koordinátákat ad vissza. Ha ez a bit be van állítva, és a felhasználó egy nyomógomb megnyomása vagy egy billentyű lenyomása helyett a pozícionálót mozgatja, akkor a **grread** függvény egy olyan listával tér vissza, amelyben az első elem az 5-ös típuskód, a második elem pedig a pozícionáló eszköz (egér vagy digitalizáló) aktuális *(X,Y)* koordinátáit jelenti. Ezt a módszert alkalmazza az AutoCAD a vontatás során.

2 (1. bit) *Minden* billentyű értékét visszaadja, beleértve a funkció- és kurzorbillentyűk kódját is, továbbá *nem* mozdítja el a kurzort egy kurzorbillentyű lenyomásakor.

4 (2. bit) A kurzornak a képernyőn történő megjelenítéséhez felhasználja a *curtype* argumentumban átadott értéket.

8 (3. bit) Nem jeleníti meg a hiba: konzol megszakítás hibaüzenetet, amikor a felhasználó lenyomja a Ctrl+C billentyűket.

curtype Az `allkeys` 2. bitjét be kell állítani, hogy a `curtype` értékek kifejthessék hatásukat.

0 Normál szálkereszt jelenik meg.

1 Nem jelenik meg kurzor (sem szálkereszt).

2 A rajzelemek kiválasztására szolgáló "célzó" kurzor jelenik meg.

Megjegyzés A `curtype` argumentumnak csak a **grread** függvényhívás idején van hatása a kurzor típusára.

Megjegyzés Az AutoCAD következő verzióiban további szabályzóbitek is definiálásra kerülhetnek.

A **grread** függvény egy listával tér vissza, melynek első eleme a bemenet típusát jelölő kód. A lista második eleme egy egész szám vagy egy pont, a bemenet típusától függően. A visszaadott értékek a következők.

A grread függvény visszatérési értékei

Első elem		*Második elem*	
Érték	*Bemenet típusa*	*Érték*	*Leírás*
2	Billentyűzet bemenet	változó	Karakterkód
3	Kiválasztott pont	háromdimenziós pont	Pont koordinátái
4	Képernyő-/Legördülő menüpont (a pozícionáló eszköztől származóan)	0-tól 999-ig 1001-től 1999-ig 2001-től 2999-ig 3001-től 3999-ig ... és így tovább, egészen 16001-től 16999-ig	Képernyőmenü-mező száma POP1 menümező száma POP2 menümező száma POP3 menümező száma ... POP16 menümező száma
5	Pozícionáló eszköz (csak ha a nyomkövetés be van kapcsolva)	háromdimenziós pont	Vontatás mód koordináták
6	BUTTONS menüpont	0-tól 999-ig 1000-től 1999-ig 2000-től 2999-ig 3000-től 3999-ig	Nyomógomb száma a BUTTONS1 menüben Nyomógomb száma a BUTTONS2 menüben Nyomógomb száma a BUTTONS3 menüben Nyomógomb száma a BUTTONS4 menüben

A grread függvény visszatérési értékei (*folytatás*)

Első elem		*Második elem*	
Érték	*Bemenet típusa*	*Érték*	*Leírás*
7	TABLET1 menüpont	0-tól 32767-ig	Táblamező száma
8	TABLET2 menüpont	0-tól 32767-ig	Táblamező száma
9	TABLET3 menüpont	0-tól 32767-ig	Táblamező száma
10	TABLET4 menüpont	0-tól 32767-ig	Táblamező száma
11	AUX menüpont	0-tól 999-ig	Nyomógomb száma az AUX1 menüben
		1000-től 1999-ig	Nyomógomb száma az AUX2 menüben
		2000-től 2999-ig	Nyomógomb száma az AUX3 menüben
		3000-től 3999-ig	Nyomógomb száma az AUX4 menüben
12	Pontkijelölő nyomógomb (6-os vagy 11-es típusú visszatérési értéket követően)	háromdimenziós pont	Pont koordinátái

A **grread** függvény végrehajtása közben az Esc billentyű lenyomása megszakítja az AutoLISP programot (hacsak az `allkeys` argumentum nem akadályozza meg ebben). Bármilyen más bemenet közvetlenül átadódik a **grread** függvénynek, lehetővé téve az alkalmazás számára az adatbeviteli eszközök feletti teljes ellenőrzést.

Amennyiben a felhasználó a képernyőmenü vagy a legördülő menü egyik mezőjén belül nyomja meg a pontkijelölő nyomógombot, a **grread** függvény a 6-os vagy 11-es típusú kóddal tér vissza, a rákövetkező hívás alkalmával viszont nem ad vissza 12-es kódot: a 12-es típusú kód csak akkor követi a 6-os vagy 11-es kódot, ha a pontkijelölő nyomógomb megnyomására a képernyő grafikus területén került sor.

Fontos, hogy a 12-es típusú adatot törölje a pufferből, mielőtt a pontkijelölő nyomógombbal vagy egy segéd-nyomógombbal újabb műveletet kísérelne meg. Ezt a következőhöz hasonló, beágyazott **grread** függvényhívással valósíthatja meg:

```
(setq code_12 (grread (setq code (grread))))
```

Ez a hívássorozat a 12-es típusú lista értékét az eszközről folyamatosan érkező bemenetként veszi át.

Megjegyzés Mivel az AutoCAD által támogatott platformokon különbözőképpen történik a bemenet kezelése, előfordulhat, hogy a **grread** függvény váratlan eredménnyel tér vissza.

- Azon platformok esetében, amelyek rendszeregeret használnak, az alapértelmezés szerinti pozícionáló eszköz 11-es kódot eredményez 6-os helyett.
- Macintosh platformon a POP menük 11-es kódot eredményeznek 4-es kód helyett. Szintén a Macintosh esetében, a dupla kattintás 11-es kódot eredményez (6-os helyett), és egy 5-ös kódú koordinátapár követi, ha az aktuális nézetablakban következett be. Ellenkezőleg, ha a dupla kattintásra nem az aktuális nézetablakban kerül sor, akkor 3-as kódú koordinátapárt eredményez, melyet egy 11-es kód követ.

grtext

Egy szöveget ír az állapotsorba vagy a képernyő menüterületeire

(grtext *[box text [highlight]]***)**

A *box* argumentum egész típusú, és megadja a helyet, ahová a szöveg kerülni fog. A *text* argumentum egy karakterlánc, amely megadja a képernyőmenüben vagy az állapotsorban megjelenítendő szöveget. A függvény csonkolja a *text* argumentumot, ha az nem fér el a rendelkezésre álló területen. A *highlight* argumentum egy egész szám, amely kiemel vagy normál módba kapcsol egy képernyőmenü helyet. Ennek az argumentumnak az értéke attól függően változik, hogy a képernyő mely területére ír a függvény. Ez a függvény a megadott szöveget jeleníti meg a menüterületen; a mögötte rejlő menüpontot nem változtatja meg. A **grtext** függvény argumentumok nélkül is meghívható, ebben az esetben az összes szöveges területet visszaállítja a szabvány állapotába. Sikeres hívás esetén a **grtext** függvény a *text* argumentumában átadott karakterláncot adja vissza, máskülönben `nil` a visszatérési érték.

Képernyőmenü-terület

A *box* argumentum pozitív vagy nulla értéke egy képernyőmenü helyet ad meg. A *box* argumentum érvényes értéke 0-tól a legmagasabb számú képernyőmenü-ablak eggyel csökkentett értékéig terjedhet. A SCREENBOXES rendszerváltozó a képernyőmenü-ablak maximális számát adja meg. Amennyiben a *highlight* argumentum pozitív egészként van megadva, a **grtext** függvény kiemelten jeleníti meg a szöveget a megjelölt ablakban. Egy ablak kiemelése automatikusan normál módba kapcsolja bármely korábban kiemelt másik ablak megjelenítését. Nulla értékű *highlight* argumentum kikapcsolja az adott menüpont kiemelését. A negatív értékű *highlight* argumentumot nem veszi figyelembe a függvény. Néhány platformon a szöveget

a *highlight* argumentum nélkül kell kiíratni, és csak utána kell kiemelni. Egy képernyőmenü hely kiemelése csak akkor lehetséges, ha a kurzor nincs az adott területen.

Állapotsor-terület

Amennyiben a **grtext** függvény hívásában a *box* értéke –1, a szöveget a képernyő állapotsorának mód részébe írja. Az állapotsor mód részének hossza megjelenítő eszközönként változik (legtöbbjük legalább 40 karaktert megenged). A következő programrészlet a **$(linelen)** DIESEL kifejezést használja a mód állapotterület hosszának a lekérdezésére.

```
(setq modelen (menucmd "M=$(linelen)"))
```

Amennyiben a *box* értéke –2, akkor a **grtext** függvény a szöveget az állapotsor koordináta részébe írja. Ha a koordináta-követés be van kapcsolva, akkor az ebbe a mezőbe írt értékek azonnal felülíródnak, amint a pozícionáló eszköztől új koordináták érkeznek. A *box* –1 vagy –2 értéke esetén a *highlight* argumentumot figyelmen kívül hagyja a függvény.

grvecs

Vektorokat rajzol a grafikus képernyőre

(grvecs *vlist [trans]*)

A *vlist* argumentum egész típusú opcionális színszámok és két pontot tartalmazó listák sorozatából álló vektor lista. A *trans* argumentum egy transzformációs mátrix, melynek segítségével megváltoztathatja a vektorlistában definiált vektorok helyét és méretarányát. Ez a mátrix lista, amely négy, egyenként négy valós számot tartalmazó listát tartalmaz.

A *vlist* formátuma a következő:

```
([szín1] ponttól1 pontig1 [szín2] ponttól pontig2 ...)
```

A színérték az őket követő összes vektorra vonatkozik, egészen addig, amíg a *vlist* egy másik színt nem határoz meg. Az AutoCAD színek száma 0-tól 255-ig terjed. Amennyiben a színérték nagyobb mint 255, a következő vektorokat a függvény *XOR tintával* rajzolja meg, mely az átrajzolt objektumok színét a kiegészítő színükre fordítja át, saját magát pedig kioltja. A szín nullánál kisebb értéke a vektor kiemelt megjelenítését eredményezi. A kiemelés módja a megjelenítő eszköztől függ. A legtöbb képernyőmeghajtó a kiemelést szaggatott vonallal jelzi, egyes típusok azonban eltérő színt használnak erre a célra.

A *ponttól* és a *pontig* pontlista-pár a vektorok végpontjait határozza meg az aktuális FKR-ben. Ezek két- vagy háromdimenziós pontok

lehetnek. Ezeket a pontokat párosával — két egymást követő pontlistában — kell megadni, máskülönben a **grvecs** hívása hibához vezet.

Az AutoCAD a vektornak a képernyőn kívülre eső részét lehagyja. Sikeres hívás esetén a **grvecs** függvény `nil` értékkel tér vissza.

A következő függvényhívás öt függőleges vonalat rajzol a grafikus képernyőre, mindegyiket más színnel:

```
(grvecs '(1 (1 2)(1 5)  vörös vonal az (1,2) ponttól az (1,5) pontig
         2 (2 2)(2 5)  sárga vonal a (2,2) ponttól a (2,5) pontig
         3 (3 2)(3 5)  zöld vonal a (3,2) ponttól a (3,5) pontig
         4 (4 2)(4 5)  v.kék vonal a (4,2) ponttól a (4,5) pontig
         5 (5 2)(5 5)  kék vonal az (5,2) ponttól az (5,5) pontig
) )
```

A következő mátrix 1.0 értékű léptéknek és 5.0,5.0,0.0 értékű eltolásnak felel meg. Amennyiben ezt a mátrixot a fenti vektorlistára alkalmazza, akkor azt 5.0,5.0,0.0 irányban tolja el.

```
'( (1.0 0.0 0.0 5.0)
   (0.0 1.0 0.0 5.0)
   (0.0 0.0 1.0 0.0)
   (0.0 0.0 0.0 1.0)
)
```

További tudnivalók

nentselp függvény, ahol a transzformációs mátrixokkal kapcsolatos további információ található

handent

Egy objektum (rajzelem) nevét adja vissza annak sorszáma alapján

(handent *handle*)

Egy elemsorszámot a *handle* argumentumban karakterláncként megadva, a **handent** függvény az aktuális szerkesztés alkalmával az adott sorszámú rajzelemhez tartozó névvel tér vissza. A **handent** függvény egyaránt visszaadja a grafikus és nem grafikus rajzelemek neveit.

Ha a rajzban a sorszámozás ki van kapcsolva, vagy a **handent** függvény érvénytelen sorszámot kapott, illetve ha nincs ilyen sorszám az aktuális rajzban, akkor a függvény `nil` értékkel tér vissza. A **handent** függvény az aktuális szerkesztés alkalmával kitörölt rajzelemek nevével is visszatérhet, így a kitörölt rajzelemek az **entdel** függvénnyel tetszés szerint visszaállíthatók.

Egy rajzelem neve szerkesztési alkalmanként változhat, de a sorszáma állandó marad. Egy adott alkalommal a következő függvényhívás

`(handent "5A2")` *lehetséges eredménye* `<Elem neve: 60004722>`

Ugyanezen rajz esetében, de másik szerkesztési alkalommal ugyanez a függvényhívás más elemnévvel térhet vissza. Az így kapott elemnév a rajzelemek kezelésére szolgáló bármely függvénnyel végzett művelet során felhasználható.

if

Feltételesen értékeli ki a kifejezéseket

(if *testexpr thenexpr [elseexpr]*)

Ha a *testexpr* értéke nem `nil`, akkor a *thenexpr* kifejezés, egyébként az *elseexpr* kifejezés kerül kiértékelésre. Az **if** függvény a kiválasztott kifejezés értékével tér vissza. Amennyiben az *elseexpr* kifejezés hiányzik és a *testexpr* értéke `nil`, akkor az **if** függvény `nil` értékkel tér vissza.

```
(if (= 1 3) "IGEN!!" "nem.") eredménye "nem."
(if (= 2 (+ 1 1)) "IGEN!!") eredménye "IGEN!!"
(if (= 2 (+ 3 4)) "IGEN!!") eredménye nil
```

További tudnivalók

progn függvény

help

Meghívja a Súgót (az összes platformon)

(help *[helpfile [topic [command]]]*)

A *helpfile* argumentum egy karakterlánc, amely megadja a súgófájl nevét. Amennyiben egy AutoCAD Súgófájlt (*.ahp*) ad meg, akkor a **help** függvény az AutoCAD Súgó megjelenítő programját használja a fájl tartalmának megjelenítésére. Amennyiben egy Windows Súgófájlt (*.hlp*) ad meg, akkor a **help** függvény a WinHelp programot használja a fájl megjelenítésére. Ha a *helpfile* argumentum egy üres karakterlánc (`""`) vagy nincs megadva, akkor az AutoCAD az alapértelmezett Súgófájlt használja. A *topic* argumentum egy kulcsszó, amely megadja a témát, amely a súgó meghívásakor jelenik meg. Amennyiben a *topic* argumentum egy üres karakterlánc (`""`), akkor a Súgó a súgófájl bevezető részét jeleníti meg. A *command* argumentum egy karakterlánc, amely megadja a Súgó ablak kezdeti állapotát.

A command argumentum értékei

Ctring	*Leírás*
HELP_CONTENTS	A Súgófájlban szereplő első témát jeleníti meg
HELP_HELPONHELP	A súgó használatáról szóló információt jelenít meg
HELP_PARTIALKEY	A Keresés párbeszédpanelt jeleníti meg a topic (téma) karakterláncként átadott kezdeti keresési szöveggel

Amennyiben Ön egy Windows Súgófájlt ad meg, a *command* argumentum a WinHelp() függvény *fuCommand* argumentuma által használt karakterlánc is lehet, amelyet WinHelp API definiál a Microsoft Windows SDK-ban.

Az egyetlen hibafeltétel, amelyet a **help** függvény visszaad az alkalmazásnak, a *helpfile* által megadott fájl jelenléte. A függvény minden más hibafeltételt egy párbeszédpanelen keresztül közöl a felhasználóval. A **help** függvény a *helpfile* karakterláncot adja vissza sikeres hívás esetén, és `nil` értékkel tér vissza, ha sikertelen volt a hívás. Amennyiben a **help** függvényt argumentum nélkül hívja meg, akkor sikeres hívás esetén egy üres karakterlánccal (`""`), egyébként `nil` értékkel tér vissza.

A következő kód meghívja a **help** függvényt, amely az *achelp.ahp* súgófájlban található SAJÁTPARANCS témáról szóló információt jeleníti meg :

```
(help "achelp.ahp" "sajátparancs")
```

További tudnivalók

"Saját súgófájlok" című résznél, a 29. oldalon további információ található az AutoCAD Súgófájl létrehozásával kapcsolatban. A **setfunhelp** a felhasználó által definiált parancshoz helyzetérzékeny súgót társít, ha a felhasználó megnyomja az F1 billentyűt.

initget

Kulcsszavakat hoz létre a következő felhasználói bevitelre szolgáló függvény számára

(initget *[bits] [string]*)

A következő függvények fogadnak kulcsszavakat: **getint**, **getreal**, **getdist**, **getangle**, **getorient**, **getpoint**, **getcorner**, **getkword**, **entsel**, **nentsel** és **nentselp**. A **getstring** függvény az egyetlen

olyan felhasználói adatbevitelre szolgáló függvény, amely nem fogad kulcsszavakat.

A `bits` argumentum egy bit-kódolású egész szám, amely megenged vagy tilt bizonyos típusú felhasználói adatbevitelt. A `string` argumentum azon kulcsszavak listáját határozza meg, amelyeket a következő adatbeviteli függvények meg fognak vizsgálni, ha a felhasználó nem a kért típusú adatot (a **`getpoint`** függvény esetében például nem egy pontot) adja meg. Amennyiben a felhasználói adatbevitel megegyezik a listában található kulcsszavak egyikével, az illető adatbeviteli függvény azzal a kulcsszóval mint karakterlánccal tér vissza. Az alkalmazás megvizsgálhatja, hogy a kulcsszavak egyikét kapta-e eredményül, és mindegyik kulcsszó esetén az annak megfelelő műveletet hajthatja végre. Ha a felhasználó által megadott adat nem a kért típusú, és egyik kulcsszóval sem egyezik meg, akkor az AutoCAD az adatbevitel megismétlését kéri a felhasználótól. Az **`initget`** által beállított bit értékek és kulcsszavak csak a következő adatbeviteli függvényhívásra vonatkoznak.

A **`initget`** függvény mindig `nil` értékkel tér vissza.

Ha az **`initget`** függvény olyan szabályzóbitet állít be, és az alkalmazás olyan felhasználói adatbevitelre szolgáló függvényt hív meg, amely esetében a bit értelmetlen, a bit egyszerűen hatástalan lesz. A bitek tetszőleges kombinációban adhatók össze, és egy 0 illetve 255 közé eső számértéket eredményezhetnek. Ha a `bits` argumentum nincs megadva, akkor az nulla (nincsenek feltételek) értékkel egyenértékű. Amennyiben a felhasználói adatbevitel nem felel meg egy vagy több megadott feltételnek (például nulla érték, amikor az nem megengedett), az AutoCAD üzenetet jelenít meg, és kéri az adatbevitel megismétlését.

Az initget függvénnyel beállítható bemenet-opciók

Bit érték	*Leírás*
1 (0. bit)	Nem engedélyezi a felhasználónak, hogy a felszólításra csupán ↵-t billentyűzzön be.
2 (1. bit)	Nem engedélyezi a felhasználónak nulla érték beírását.
4 (2. bit)	Nem engedélyezi a felhasználónak, hogy negatív értéket írjon be.
8 (3. bit)	Megengedi a felhasználónak, hogy az aktuális rajz határain kívüli pontot adjon meg. Ez a feltétel vonatkozik a következő felhasználói adatbevitelre szolgáló függvényre, még akkor is, ha a LIMCHECK rendszerváltozó jelenleg be van kapcsolva.
16 (4. bit)	(Jelenleg használaton kívül)

Az initget függvénnyel beállítható bemenet-opciók (*folytatás*)

Bit érték	*Leírás*
32 (5. bit)	Szaggatott vonalat használ a gumivonal vagy az ablak rajzolásához. Azon függvények esetében, amelyek megengedik a felhasználónak, hogy a grafikus képernyő egy helyének kiválasztásával adjon meg egy pontot, ez a bit azt eredményezi, hogy a gumivonal vagy a keret tömör vonal helyett szaggatottként jelenik meg. (Egyes megjelenítő vezérlők eltérő színt használnak szaggatott vonal helyett.) Ha a POPUPS rendszerváltozó értéke 0, az AutoCAD figyelmen kívül hagyja ezt a bitet.
64 (6. bit)	Nem engedélyezi a *Z* koordináta bevitelét a `getdist` függvény számára; biztosítja az alkalmazás számára, hogy egy kétdimenziós távolsággal térjen vissza.
128 (7. bit)	Tetszőleges bevitelt engedélyez, mintha az egy kulcsszó lenne, bármely más szabályzóbit vagy kulcsszó elfogadásával. Ez a bit megelőzi a 0. bitet; ha a 7. és 0. bit be van állítva, és a felhasználó az ↵ billentyűt üti le, akkot üres karkterláncot ad vissza.

Megjegyzés Az AutoLISP jövőbeni verziói további **`initget`** szabályzóbiteket is használhatnak, ezért kerülje a táblázatban nem található, illetve az ebben a részben nem ismertetett bitek beállítását.

A speciális szabályzóértékeket csak azok a **`getxxx`** függvények veszik figyelembe, amelyeknél azoknak értelmük van.

A felhasználói bevitelre szolgáló függvények és az alkalmazható szabályzóbitek

Függvény	*Figyelembe vesz kulcsszavakat*	*Szabályzóbitek értékei*						
		Nem üres (1)	*Nem nulla (2)*	*Nem negatív (4)*	*Határok nélküli (8)*	*Szaggatott vonalak (32)*	*kétdimenziós távolság (64)*	*Tetszőleges bemenet (128)*
`getint`	•	•	•	•				•
`getreal`	•	•	•	•				•
`getdist`	•	•	•	•		•	•	•
`getangle`	•	•	•			•		•
`getorient`	•	•	•			•		•
`getpoint`	•	•			•	•		•
`getcorner`	•	•			•	•		•

A felhasználói bevitelre szolgáló függvények és az alkalmazható szabályzóbitek

Függvény	*Figyelembe vesz kulcsszavakat*	*Szabályzóbitek értékei*						
		Nem üres (1)	*Nem nulla (2)*	*Nem negatív (4)*	*Határok nélküli (8)*	*Szaggatott vonalak (32)*	*kétdimenziós távolság (64)*	*Tetszőleges bemenet (128)*
`getkword`	•	•						•
`entsel`	•							
`nentsel`	•							
`nentselp`	•							

Kulcsszavak meghatározása

A *`string`* argumentum értelmezése a következő szabályok szerint történik:

Minden egyes kulcsszót egy vagy több szóköz választ el egymástól. Például a `"Szélesség Magasság Mélység"` karakterlánc három kulcsszót határoz meg.

Minden kulcsszó csak betűket, számokat és kötőjeleket(-) tartalmazhat.

Két módszer létezik a kulcsszavak rövidítésére:

- A kulcsszó kívánt részét nagybetűkkel, a fennmaradó részt pedig kisbetűkkel adja meg. A nagybetűs rövidítés a kulcsszó bármely részén szerepelhet (például, `"VTípus"`, `"kiLép"` vagy `"fenT"`).
- A *teljes* kulcsszót nagybetűkkel adja meg, és közvetlenül utána, egy vesszővel elválasztva megismétli a kívánt karaktereket (például, `"VTÍPUS,VT"`). Ebben az esetben a rövidítésnek tartalmaznia kell a kulcsszó első betűjét, ami azt jelenti, hogy a `"KILÉP,L"` meghatározás nem érvényes.

A két rövid példában szereplő meghatározások , `"VTípus"` és `"VTÍPUS,VT"`, egyenértékűek: a kulcsszó azonosításához elegendő, ha a felhasználó a **VT** karaktereket billentyűzi be (akár nagy-, akár kisbetűvel). A felhasználó a szükséges rész *után* következő karaktereket is beírhatja, ha azok nem mondanak ellent a meghatározásnak. A fenti példánál maradva, a felhasználó **VTÍ** vagy **VTÍP** választ is bebillentyűzhet, **V** azonban nem elegendő.

Ha a *`string`* argumentumban a kulcsszavak teljes egészében nagybetűvel *vagy* kisbetűvel szerepelnek, és nem követi őket a vessző után a kívánt rész, akkor azokat az AutoCAD csak egészben bebillentyűzve ismeri fel.

További tudnivalók

"A felhasználói bevitelre szolgáló függvények feltételeinek szabályozása" című részнél, a 235. oldalon

inters

Megtalálja két vonal metszéspontját

(inters *pt1 pt2 pt3 pt4 [onseg]*)

A *pt1* és *pt2* az első, a *pt3* és *pt4* a második egyenes végpontjai. Ha az *onseg* argumentum jelen van, és értéke `nil`, akkor a négy *pt* argumentum által meghatározott egyeneseket az **inters** függvény végtelen hosszúságúnak tekinti, és metszéspontjukat abban az esetben is kiszámítja, ha az egyik, vagy mindkét egyenes végpontján túl helyezkedik el. Ha az *onseg* argumentumot elhagyja, vagy értéke nem `nil`, akkor a metszéspontnak a két egyenes közös pontjának kell lennie, máskülönben az **inters** függvény `nil` értékkel tér vissza. Az **inters** függvény `nil` értékkel tér vissza, ha a két vonal nem metszi egymást.

Minden pont az aktuális FKR-ben van kifejezve. Amennyiben mind a négy argumentum térbeli pont, az **inters** függvény háromdimenziós metszéspontra vizsgálja meg az egyeneseket. Ha a pontok bármelyike kétdimenziós pont, az **inters** függvény az aktuális szerkesztésősíkra vetíti őket, és csak a vetületük síkbeli metszéspontját keresi.

```
(setq a '(1.0 1.0) b '(9.0 9.0))
(setq c '(4.0 1.0) d '(4.0 2.0))
(inters a b c d)          eredménye  nil
(inters a b c d T)        eredménye  nil
(inters a b c d nil)      eredménye  (4.0 4.0)
```

itoa

Egy egész szám karakterlánccá konvertált értékével tér vissza

(itoa *int*)

Az *int* argumentum egy egész számot ad meg.

```
(itoa 33)                 eredménye  "33"
(itoa -17)                eredménye  "-17"
```

lambda

Egy névtelen függvényt definiál

(lambda *arguments expr...*)

Akkor használja a **lambda** függvényt, ha új függvény definiálása nem indokolt. A programozó szándékát is világossá teszi, ha a függvény definíciója a felhasználás helyén található. Ez a függvény az utolsó *expr* értékével tér vissza, és gyakran használatos az **apply** és/vagy a **mapcar** függvényekkel együtt, amikor egy listára alkalmazzuk őket.

```
(apply '(lambda (x y z)
          (* x (- y z))
        )
        '(5 20 14)
)                              eredménye  30
```

és

```
(setq counter 0)
(mapcar (lambda (x)
          (setq counter (1+ counter))
          (* x 5)
        )
        '(2 4 -6 10.2)
)                              eredménye  (10 20 -30 51.0)
```

last

Egy lista utolsó elemével tér vissza

(last *list*)

```
(last '(a b c d e))        eredménye  E
(last '(a b c (d e)))      eredménye  (D E)
```

Mint azt a példa mutatja, a **last** függvény atommal és listával is visszatérhet.

Megjegyzés Első pillantásra a **last** megfelelő függvénynek tűnik, arra, hogy megkapjuk egy pont *Y* koordinátáját. Ez igaz is kétdimenziós pontok (két valós számból álló listák) esetében, de a háromdimenziós pontok esetében a **last** függvény a *Z* koordinátát fogja eredményül adni. Amennyiben azt szeretné, hogy programja két- vagy háromdimenziós pontok esetében is helyesen működjön, az *Y* koordináták elérésére a **cadr**, a *Z* koordináták elérésére pedig a **caddr** függvényt használja.

length

Egy listában szereplő elemek számát adja eredményül

(length *lst*)

```
(length '(a b c d))          eredménye 4
(length '(a b (c d)))        eredménye 3
(length '())                 eredménye 0
```

list

Listát hoz létre tetszőleges számú kifejezésből

(list *expr...*)

```
(list 'a 'b 'c)              eredménye (A B C)
(list 'a '(b c) 'd)          eredménye (A (B C) D)
(list 3.9 6.7)               eredménye (3.9 6.7)
```

Az AutoLISP programnyelvben gyakran használjuk ezt a függvényt két- vagy háromdimenziós pontok (két illetve három valós elemű listák) definiálására.

Amennyiben a listában nincsenek változók vagy definiálatlan tételek, a **list** függvény helyett közvetlenül a **quote** függvénnyel is megadhatja a listát. A quote függvény jele az aposztróf (**'**) .

`'(3.9 6.7)` *ugyanazt jelenti, mint* `(list 3.9 6.7)`

Ez használható asszociációs listák létrehozására és pontok definiálására is.

További tudnivalók

quote függvény

listp

Megvizsgálja, hogy a tétel lista-e

(listp *item*)

Amennyiben az *item* egy lista, akkor `T`, egyébként `nil` értékkel tér vissza.

```
(listp '(a b c))             eredménye T
(listp 'a)                   eredménye nil
(listp 4.343)                eredménye nil
```

Megjegyzés Mivel a `nil` atom és lista is egyben, a **`listp`** függvény `T` értékkel tér vissza, ha az argumentuma `nil`.

```
(listp nil)                    eredménye  T
```

load_dialog

Betölt egy DCL fájlt

`(load_dialog` ***`dclfile`*****`)`**

A *`dclfile`* argumentum egy karakterlánc, amely a betöltendő DCL fájlt adja meg. Amennyiben a *`dclfile`* argumentum nem ad meg fájlkiterjesztést, akkor a *.dcl* kiterjesztést veszi alapértelmezettnek. Sikeres hívás esetén egy pozitív egésszel (`dcl_id`) tér vissza. Ha nem tudja megnyitni a fájlt, akkor negatív egésszel tér vissza. A `dcl_id` értéket mint sorszámot használja fel az ezt követő **`new_dialog`** és **`unload_dialog`** függvényhívásoknál.

A **`load_dialog`** függvény a fájlokat az AutoCAD könyvtárkeresési útvonal szerint keresi.

Ez a függvény az **`unload_dialog`** függvény ellentettje. Az alkalmazás többszöri **`load_dialog`** hívással több DCL fájlt tölthet be.

load

Egy fájl AutoLISP kifejezéseit értékeli ki

`(load` ***`filename [onfailure]`*****`)`**

A *`filename`* argumentum egy karakterlánc, amely megadja a fájlnevet. Amennyiben a *`filename`* argumentum nem tartalmazza a kiterjesztést, akkor a függvény a *.lsp* kiterjesztést feltételezi. A **`load`** függvény sikertelen hívás esetén az *`onfailure`* argumentum értékével tér vissza. Azonban ha az *`onfailure`* nincs megadva, akkor a **`load`** sikertelen hívása AutoLISP hibát okoz. Ha a művelet sikeres, akkor a **`load`** a fájlban található kifejezés értékével tér vissza.

A *`filename`* könyvtár előtagot is tartalmazhat, mint például *"/funkc/teszt1"*. DOS rendszerek esetében a lemezmeghajtó betűjelének használata is megengedett. Törtjelet (`/`) vagy két visszahajló törtjelet (`\\`) is használhat a könyvtárak elválasztására. Abban az esetben, ha a *`filename`* karakterláncba nem foglalja bele a könyvtár előtagot, a **`load`** függvény az AutoCAD könyvtárútvonalában keresi az adott fájlt. Ha a fájlt megtalálja az útvonalon, akkor a **`load`** betölti azt.

Amennyiben az **`onfailure`** argumentum érvényes AutoLISP függvény, akkor kiértékelésre kerül. A legtöbb esetben az *`onfailure`* argumentumnak karakterláncnak vagy atomnak kell lennie. Ez

lehetővé teszi, hogy a **load** függvényt hívó AutoLISP alkalmazás hiba esetén alternatív műveletet hajtson végre.

Példaként tételezzük fel, hogy a */fred/teszt1.lsp* fájl a következőket tartalmazza:

```
(defun SAJÁT-FÜGGV1 (x)
   ... a függvény ...
)
(defun SAJÁT-FÜGGV2 (x)
   ... a függvény ...
```

valamint, hogy a *teszt2.lsp* fájl nem létezik. Ebben az esetben

```
(load "/fred/teszt1")           eredménye  SAJÁT-FÜGGV2
(load "\\fred\\teszt1")         eredménye  SAJÁT-FÜGGV2
(load "/fred/teszt1" "rossz")   eredménye  SAJÁT-FÜGGV2
(load "teszt2" "rossz")         eredménye  "rossz"
(load "teszt2")                 AutoLISP hibát okoz
```

A **load** függvényt egy másik AutoLISP függvényen belül is használhatja, sőt rekurzívan is hívhatja (az éppen betöltött fájlban).

További tudnivalók

defun függvény és "Szimbólum- és függvénykezelés" című résznél, a 210. oldalon

log

Egy szám természetes alapú logaritmusát adja vissza valós számérték formájában

(log *num*)

```
(log 4.5)    eredménye  1.50408
(log 1.22)   eredménye  0.198851
```

logand

A számok listáján elvégzett bitenkénti logikai AND művelet eredményével tér vissza

(logand *int int...*)

```
(logand 7 15 3)   eredménye  3
(logand 2 3 15)   eredménye  2
(logand 8 3 4)    eredménye  0
```

logior

A számok listáján elvégzett bitenkénti logikai inkluzív OR művelet eredményével tér vissza

(logior *int int...*)

```
(logior 1 2 4)          eredménye 7
(logior 9 3)            eredménye 11
```

lsh

Egy szám megadott számú bittel történő, bitenkénti logikai eltolásával nyert értékét adja vissza

(lsh *int numbits*)

A pozitív `numbits` az `int` argumentum balra tolását, a negatív pedig jobbra tolását jelenti. Az eltolás mindkét esetben a nullák beillesztését, és a túlcsorduló bitek elvesztését jelenti. Amennyiben a szám legmagasabb helyiértékű bitjére (31. bit) nulla kerül, a visszatérési érték pozitív, különben pedig negatív lesz.

```
(lsh 2 1)               eredménye 4
(lsh 2 -1)              eredménye 1
(lsh 40 2)              eredménye 160
```

mapcar

Egy függvény végrehajtásakor kapott listával tér vissza, oly módon, hogy a lista vagy listák elemeit egyenként a függvény argumentumaiként kezeli

(mapcar *function list1... listn*)

A listák számának egyeznie kell a `function` által igényelt argumentumok számával.

```
(setq a 10 b 20 c 30)
(mapcar '1+ (list a b c)) eredménye (11 21 31)
```

Ez megfelel az alábbiaknak:

```
(1+ a)
(1+ b)
(1+ c)
```

azzal a különbséggel, hogy a **mapcar** az eredményt lista formájában adja vissza.

A **lambda** függvénnyel megadhat olyan névtelen függvényt, amelyet a **mapcar** függvény hajt végre. Ez hasznos lehet abban az esetben, amikor a függvény argumentumai közül néhány érték konstans, vagy más forrásból származik.

```
(mapcar '(lambda (x)
          (+ x 3)
         )
        '(10 20 30)
)                          eredménye (13 23 33)
```

max

Az adott számok közül a legnagyobbat adja vissza

(max *number number...*)

```
(max 4.07 -144)          eredménye 4.07
(max -88 19 5 2)         eredménye 19
(max 2.1 4 8)            eredménye 8.0
```

mem

Kijelzi az AutoLISP memóriájának aktuális állapotát

(mem)

Kijelzi az AutoLISP memóriájának aktuális állapotát, és `nil` értékkel tér vissza. Az alábbi információkat jeleníti meg:

A *Csomópontok* mező az eddig lefoglalt csomópontok teljes számát mutatja, amelynek egyenlőnek kell lennie a szegmensek méretének és számának szorzatával.

A *Szabad csomópontok* mező a jelenleg *szabad* listán található csomópontok számát mutatja, melyek hulladékgyűjtési folyamat eredményeképpen kerültek a listára.

A *Szegmensek* mező a kijelölt csomóponti szegmensek számát jelöli.

Az *Allokáció* mező az aktuális szegmensméretet mutatja .

A Gyűjtések mező az akár automatikusan, akár külön utasításra végrehajtott hulladékgyűjtő folyamatok számát mutatja.

További tudnivalók

15., "Memóriakezelés" című fejezet

member

Egy kifejezés előfordulását keresi egy listában, és a listának a kifejezés első előfordulásával kezdődő maradékát adja vissza

(member *expr list*)

Amennyiben az *expr* kifejezés nem fordul elő a *list* argumentumban, akkor a **member** függvény `nil` értékkel tér vissza.

```
(member 'c '(a b c d e)) eredménye (C D E)
(member 'q '(a b c d e)) eredménye nil
```

menucmd

Menü parancsokat ad ki, vagy beállítja és visszaadja a menüpont státuszát

(menucmd *string*)

A *string* argumentum egy karakterlánc, amely a menüterületet és a hozzárendelt értéket adja meg. A *string* argumentum a következő paraméterekkel rendelkezik.

"*menü_terület=érték*"

DOS és UNIX rendszerek esetében a *menu_terület* lehetséges értékei megegyeznek a menüfájl almenü-hivatkozásaiban megengedett értékekkel. További információ az "Almenük" című részнél, a 91. oldalon és a "Hivatkozás egy legördülő vagy kurzor-almenüre" című részнél, a 117. oldalon található.

Menü_terület karakterlánc értékei

Menü_terület karakterlánc	*Menüszakasz*
B1–B4	BUTTONS menük 1-től 4-ig
A1–A4	AUX menük 1-től 4-ig
P0–P16	Legördülő (POP) menük 0-tól 16-ig
I	Ábrazóna menük
S	SCREEN menük
T1–T4	TABLET menük 1-től 4-ig
M	DIESEL karakterlánc-kifejezések

Windows operációs rendszer esetében a *menü_terület* számára a DOS által megengedett értékek a *Gmenugroup.nametag* szintaxissal egészülnek ki (További információ a "Windows-specifikus menüjellemzők" című résznél, a 127. oldalon található).

Az *érték* paraméter a *menü_terület* paraméterhez rendelt értéket adja meg.

A **menucmd** függvény képes átkapcsolni az AutoCAD menü egyes lapjai között és menüket is megjeleníthet. Ez lehetővé teszi az AutoLISP programok számára az ábramenük használatát, és olyan más menük megjelenítését, amelyekből a felhasználó választhat. Az AutoLISP programok engedélyezhetnek, letilthatnak és megjelölhetnek menüpontokat.

A következő programrészlet a TÖBBIKON nevű ábrazóna-menüt jeleníti meg.

```
(menucmd "I=többikon")    Betölti a TÖBBIKON ábrazóna-menüt
(menucmd "I=*")           Megjeleníti a menüt
```

Az alábbi programrészlet ellenőrzi a legördülő POP11 menü harmadik menüpontjának státuszát. Amennyiben a menüpont jelenleg engedélyezett, a **menucmd** függvény letiltja azt.

```
(setq s (menucmd "P11.3=?"))  Lekérdezi a menüpont státuszát
(if (= s "")                  Ha a státusz egy üres karakterlánc,
  (menucmd "P11.3=~")         akkor letiltja a menüpontot
)
```

Az előző programrészlet nem megbízható. Az engedélyezésen és letiltáson kívül a menüpont írásjeleket is fogadhat. A `(menucmd "P11.3=?")` kód eredménye `"!."` is lehet, ami azt jelenti, hogy a menüpont jelenleg ellenőrzés alatt áll. Ez a kód feltételezné, hogy a menüpont le van tiltva, és a menüpont letiltása nélkül folytatná működését. Ha a kód tartalmazna egy **wcmatch** függvényhívást, akkor az ellenőrizné a (~) karakter jelenlétét, és ennek megfelelően folytatná működését.

A **menucmd** függvény lehetővé teszi az AutoLISP programok számára, hogy kihasználják a DIESEL karakteres kifejezésnyelv előnyeit. Bizonyos dolgokat sokkal könnyebb DIESEL nyelven megvalósítani, mint annak AutoLISP megfelelőjével. A következő példa az aktuális napot és dátumot adja vissza karakterláncként:

```
(menucmd "M=$(edtime,$(getvar,date),DDDD\",\" D MONTH YYYY)")
         eredménye   "Vasárnap, 16 Július 1995"
```

További tudnivalók

A 6., "Programozási felületek" című fejezet az AutoLISP menücím státusz elérésével, és az 5., "Az állapotsor konfigurálása és a DIESEL

karakteres kifejezésnyelv" című fejezet a DIESEL nyelv használatával kapcsolatos információkat tartalmazza

min

Az adott számok közül a legkisebbet adja vissza

(min *number number...*)

```
(min 683 -10.0)          eredménye -10.0
(min 73 2 48 5)          eredménye  2
(min 2 4 6.7)            eredménye  2.0
```

minusp

Megvizsgálja, hogy a szám negatív-e

(minusp *num*)

Amennyiben *number* negatív, akkot `T`, egyébként pedig `nil` a visszatérési érték.

```
(minusp -1)              eredménye T
(minusp -4.293)          eredménye T
(minusp 830.2)           eredménye nil
```

mode_tile

Beállítja a párbeszédpanel-zóna állapotát

```
(mode_tile key mode)
```

A *key* argumentum egy olyan karakterlánc, amely megadja a zónát, és amely megkülönbözteti a kis- és nagybetűket. A *mode* argumentum egész típusú; értékeit az alábbi táblázat tartalmazza.

Mode argumentum értékei

Érték	*Leírás*
0	Zóna engedélyezése
1	Zóna letiltása
2	A fókusz ráirányítása a zónára
3	A szerkesztőablak tartalmának kiválasztása
4	Az ábra kiemelésének be- vagy kikapcsolása

namedobjdict

Az aktuális rajz megnevezett objektum szótárának rajzelem nevét adja vissza. Ez a név a rajzban szereplő összes nem grafikus objektum kiindulópontjául szolgál

```
(namedobjdict)
```

A visszaadott név és a szótár hozzáférő függvények segítségével az alkalmazás hozzáfér a rajz nem grafikus elemeihez.

nentsel

Arra kéri a felhasználót, hogy egy pont megadásával válasszon ki egy objektumot (rajzelemet), és hozzáférést biztosít egy összetett objektum definíciós adataihoz

```
(nentsel [prompt])
```

A *prompt* argumentum egy karakterlánc, amely promptként fog megjelenni. Amennyiben elmarad, akkor a Válasszon objektumokat prompt jelenik meg.

A **nentsel** függvény kéri a felhasználót, hogy válasszon ki egy objektumot. A Traszter aktuális beállításait nem veszi figyelembe, hacsak a felhasználó ezt külön nem kéri. További szolgáltatásként a Parancs prompt után, a **nentsel** függvény az előző **initget** hívás által meghatározott kulcsszavakat is elfogadja.

Amennyiben a választott objektum nem összetett rajzelem (nem vonallánc vagy blokk), a **nentsel** az **entsel** függvénnyel azonos értékkel tér vissza. Ha azonban a kiválasztott rajzelem vonallánc, akkor a **nentsel** által visszaadott lista az alárendelt rajzelem (csomópont) nevét és a kiválasztás pontját tartalmazza. Ez hasonlít az **entsel** visszatérési listájához, azzal a különbséggel, hogy a vonallánc fejrésze helyén a kiválasztott csomópont neve szerepel. A **nentsel** függvény mindig a vonallánc kiválasztott szakaszának kezdőpontjával tér vissza. Például, a vonallánc harmadik szakaszának kiválasztásakor ez a harmadik csomópont lesz. A **nentsel** soha nem adja vissza egy vonallánc Seqend alárendelt rajzelemét.

Egy blokkreferencián belüli attribútum kiválasztásakor a függvény az attribútum nevével és a kiválasztott ponttal tér vissza. Amennyiben a kiválasztott objektum nem a blokkreferencia attribútuma, akkor a **nentsel** négy elemből álló listával tér vissza.

Egy blokkon belüli objektum kiválasztásakor a visszaadott lista első eleme a kiválasztott rajzelem neve. A második elem az objektum kiválasztásához használt pont koordinátáit tartalmazó lista.

A harmadik elem neve Modell-Világ transzformációs mátrix. Ez egy olyan lista, amely négy -- koordinátacsoportokat tartalmazó -- allistából áll. Ennek a mátrixnak a segítségével lehet egy rajzelem definíciós adatait egy belső Modell koordináta-rendszerből (MKR) a Világ koordináta-rendszerbe (VKR) transzformálni. Az MKR origóját a kiválasztott elemet tartalmazó blokk beillesztési pontja határozza meg. Tengelyeinek irányát a blokk létrehozásakor használt FKR tengelyeinek iránya határozza meg.

A negyedik elem egy lista, amely a kiválasztott objektumot magában foglaló blokk elemnevét tartalmazza. Amennyiben a kiválasztott rajzelem egy beágyazott blokk (blokk a blokkban) része, akkor a lista kiegészül mindazon blokkok rajzelem nevével, amelyekbe a kiválasztott rajzelem beágyazódott, a legbelső blokktól kezdve egészen a rajzba beillesztett blokkig.

```
(<Elem neve: enév1>        A rajzelem neve

  (Px Py Pz)               Kijelölési pont
  ( (X0 Y0 Z0)             Modell-Világ transzformációs mátrix
    (X1 Y1 Z1)
    (X2 Y2 Z2)
    (X3 Y3 Z3)
  )
```

```
  (<Elem neve:enév2>        A kiválasztott rajzelemet tartalmazó
     .                      legmélyebben beágyazott blokk neve
     .
     .
   <Elem neve:enévn>)       A kiválasztott rajzelemet tartalmazó
)                           legkülső blokk neve
```

Miután rendelkezésére áll a rajzelem neve és a Modell-Világ transzformációs mátrix, transzformálni tudja az elemdefiníciós adatok pontjait az MKR-ből a VKR-be. Az MKR-koordinátákban kifejezett definíciós pontokat úgy kaphatja meg, ha az **entget** és az **assoc** függvényeket alkalmazza a rajzelem nevére. A **nentsel** által szolgáltatott Modell-Világ transzformációs mátrix ugyanazt a célt szolgálja, mint az, amelyet a **nentselp** függvény ad eredményül. Ez azonban egy 4x3-as —négy pontból álló tömbként átadódó—mátrix, amelyben megállapodás szerint a sorok képviselnek egy-egy pontot, nem pedig az oszlopok. A transzformáció a következő mátrixszorzással írható le:

$$\begin{bmatrix} X' & Y' & Z' & 1.0 \end{bmatrix} = \begin{bmatrix} X & Y & Z & 1.0 \end{bmatrix} \cdot \begin{vmatrix} M_{00} & M_{01} & M_{02} \\ M_{10} & M_{11} & M_{12} \\ M_{20} & M_{21} & M_{22} \\ M_{30} & M_{31} & M_{32} \end{vmatrix}$$

Ennek alapján az új koordináták a következő egyenletekkel származtathatók :

$$X' = XM_{00} + YM_{10} + ZM_{20} + M_{30}$$

$$Y''''' = XM_{01} + YM_{11} + ZM_{21} + M_{31}$$

$$Z' = XM_{02} + YM_{12} + ZM_{22} + M_{32}$$

Az *Mij* értékek -- ahol $0 \leq i, j \leq 2$ -- a Modell-Világ transzformációs mátrix koordinátái, *X*, *Y* és *Z* az elemdefiníciós adatpontot kifejező MKR koordináták, *X'*, *Y'* és *Z'* pedig az eredményül kapott elemdefiníciós adatpontot kifejező VKR koordináták.

Megjegyzés Egyedül ez az AutoLISP függvény használ ilyen típusú mátrixot. A **nentselp** függvény ahhoz hasonló mátrixot ad eredményül, mint amilyet más AutoLISP és ADS függvények használnak.

További tudnivalók

"Rajzelemnév függvények" című résznél, a 273. oldalon és **entsel** illetve **initget** függvények

nentselp

Hasonló a nentsel függvényhez, azonban itt nincs szükség felhasználói adatbevitelre

(nentselp *[prompt] [pt]***)**

Az opcionális *prompt* argumentum mellett, a **nentselp** függvény ugyancsak opcionális argumentumként (*pt*) egy kiválasztási pont megadását is elfogadja. Ez lehetővé teszi egy objektum kiválasztását felhasználói adatbevitel nélkül is. A **nentselp** függvény a következőképpen definiált 4x4-es transzformációs mátrixot szolgáltatja:

$$\begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{bmatrix}$$

A mátrix első három oszlopa a léptéket és a forgatást határozza meg. A negyedik oszlop eltolási vektor.

Az ilyen típusú mátrixot használó függvények négydimenziós oszlopvektorként kezelik a pontokat. A pontot *homogén koordinátákkal* fejezik ki, úgy, hogy a pontvektor negyedik eleme egy *arányossági tényező*, amelynek értéke normális esetben 1.0. A mátrix utolsó sora az [M_{30} M_{31} M_{32} M_{33}] vektor, amelynek névleges értéke [0 0 0 1]; ezt a vektort az ilyen típusú mátrixot használó függvények jelenleg nem veszik figyelembe. A megállapodás szerint egy pont transzformálását a következő alakú mátrixszorzás jelenti:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1.0 \end{bmatrix} = \begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1.0 \end{bmatrix}$$

Ez a szorzás a pont egyes koordinátáit a következőképpen adja meg:

$$X' = XM_{00} + YM_{01} + ZM_{02} + M_{03}\,(1.0)$$

$$Y''' = XM_{10} + YM_{11} + ZM_{12} + M_{13}\,(1.0)$$

$$Z' = XM_{20} + YM_{21} + ZM_{22} + M_{23}\,(1.0)$$

Amint a fenti egyenletekből látható, az arányossági tényező és a mátrix utolsó sora elhanyagolható, az eredményt nem befolyásolja.

new_dialog

Egy új párbeszédpanelt indít, megjeleníti azt, és egy alapértelmezés szerinti akciót rendelhet hozzá

(new_dialog *dlgname dcl_id [action [screen-pt]]*)

A *dlgname* argumentum egy karakterlánc, amely megadja a párbeszédpanelt, a *dcl_id* pedig azonosítja a DCL fájlt (ennek az értéknek a **load_dialog** hívásból kell származnia).

Az *action* argumentum, amelyet meg kell adnia, ha a *screen-pt* argumentumot megadja, egy karakterlánc, amely az alapértelmezés szerinti akciónak szánt AutoLISP kifejezést tartalmazza. Amennyiben nem kíván alapértelmezés szerinti akciót definiálni, akkor az *action* argumentumnak adjon értékül egy üres karakterláncot(""). A *screen-pt* argumentum egy kétdimenziós pontlista, amely a párbeszédpanelnek az *X,Y* pozícióját adja meg a képernyőn. A pont általában a párbeszédpanel bal felső sarkát jelöli, ez azonban platformtól függő, csakúgy, mint az adott rendszerben használt hosszegység a pozíció megadására. Amennyiben ennek a pontnak a helyén a '(-1 -1) listát adja át, a párbeszédpanel az alapértelmezés szerinti helyen (az AutoCAD grafikus képernyőjének közepén) jelenik meg.

A **new_dialog** sikeres hívás esetén T; máskülönben nil értékkel tér vissza.

Az alkalmazásában a **new_dialog** függvényt a **start_dialog** előtt kell meghívnia. A párbeszédpanel teljes inicializálására, vagyis a zónák értékének a beállítására, az ábrák és a listaablakok listáinak a megszerkesztésére, valamint az akcióknak az adott zónákhoz való rendelésére (az **action_tile** útján), a **new_dialog** hívása után, és a **start_dialog** hívása előtt kell hogy sor kerüljön.

Az alapértelmezés szerinti akció minden alkalommal kiértékelésre kerül, amikor a felhasználó egy olyan aktív zónát jelöl ki, melyhez explicit módon -- az **action_tile** útján vagy a DCL fájlban -- nincs akció vagy visszajelzési függvény rendelve.

Megjegyzés Mindig vizsgálja meg a **new_dialog** által visszaadott státuszt. Ha a **new_dialog** sikertelen hívása után kísérli meg a **start_dialog** hívását, az előre nem látható eredménnyel járhat.

not

Megvizsgálja, hogy egy tétel nil értékű-e

(not *item*)

Amennyiben az *item* értéke nil, akkor T, egyébként nil a visszatérési érték.

```
(setq a 123 b "string" c nil)
(not a)                    eredménye  nil
(not b)                    eredménye  nil
(not c)                    eredménye  T
(not '())                  eredménye  T
```

Rendszerint a **null** függvény használatos a listák, a **not** pedig minden egyéb adattípus vizsgálatára, néhány más típusú vezérlőfüggvénnyel együtt.

nth

Egy lista n-edik elemét adja vissza

(nth *n list*)

Az *n* argumentum a keresett elem sorszáma (a lista első elemének sorszáma nulla). Amennyiben az *n* nagyobb, mint a *list* utolsó elemének sorszáma, akkor az **nth** nil értékkel tér vissza.

```
(nth 3 '(a b c d e))       eredménye  D
(nth 0 '(a b c d e))       eredménye  A
(nth 5 '(a b c d e))       eredménye  nil
```

null

Megvizsgálja, hogy a tételhez nil érték tartozik-e

(null *item*)

Amennyiben az *item* argumentumhoz rendelt érték nil, akkor T, egyébként pedig nil a visszatérési érték.

```
(setq a 123 b "karakterek" c nil)
(null a)                    eredménye  nil
(null b)                    eredménye  nil
(null c)                    eredménye  T
(null '())                  eredménye  T
```

numberp

Megvizsgálja, hogy a tétel egy valós vagy egész szám-e

(numberp *item*)

Amennyiben az *item* argumentum egész vagy valós szám, akkor T, egyébként pedig nil a visszatérési érték.

```
(setq a 123 b 'a)
(numberp 4)                 eredménye  T
(numberp 3.8348)            eredménye  T
(numberp "Hogyvagy")        eredménye  nil
(numberp a)                 eredménye  T
(numberp b)                 eredménye  nil
(numberp (eval b))          eredménye  T
```

open

Megnyit egy fájlt az AutoLISP I/O függvényei általi hozzáférésre

(open *filename mode*)

A *filename* argumentum egy olyan karakterlánc, amely a megnyitni kívánt fájl nevét és kiterjesztését adja meg . A *mode* argumentum azt az írás/olvasás flaget tartalmazza, amely csak egyetlen kisbetűből álló karakterlánc lehet. A módok érvényes betűjeleit a következő táblázatban foglaltuk össze.

Az open függvény mód opciói

Open mód	*Leírás*
"r"	Olvasásra nyitja meg a fájlt. Ha a *filename* nem létezik, az **open** függvény nil értékkel tér vissza.
"w"	Írásra nyitja meg a fájlt. Ha a *filename* nem létezik, akkor a függvény egy új fájlt hoz létre, és megnyitja azt. Ha a *filename* nevű fájl már létezik, a benne levő adatok felülíródnak .

Az open függvény mód opciói (*folytatás*)

Open mód	*Leírás*
"a"	Hozzáfűzésre nyitja meg a fájlt. Ha *filename* nem létezik, akkor a függvény egy új fájlt hoz létre, és megnyitja azt. Ha a *filename* nevű fájl már létezik, akkor ezt a függvény megnyitja, és a fájlban lévő adatok végére áll; így a fájlba írt új adatok a meglévő adatokhoz lesznek hozzáfűzve.

Megjegyzés DOS operációs rendszerek esetében néhány program, illetve szövegszerkesztő a szövegfájlok végére egy "fájl vége" jelet (CTRL Z, decimális ASCII kódja 26) illeszt. Ha a DOS a szövegfájlok olvasásakor, egy CTRL Z jellel találkozik, akkor egy fájl vége státuszt küld, még abban az esetben is, ha ezt a jelet további adatok követik. Amennyiben az OPEN függvény "a" módját használva adatokat kíván egy másik programmal készített fájl végéhez hozzáfűzni, először bizonyosodjon meg arról, hogy a szóbanforgó program nem illeszt CTRL Z jelet a szövegfájl végéhez.

Az **open** függvény olyan fájlleíróval tér vissza, amelyet az egyéb I/O függvények fognak felhasználni. A fájlleírót olyan szimbólumhoz kell rendelni, amely **setq** függvényt használja.

```
(setq a (open "fájl.kit" "r"))
```

Tételezzük fel, hogya következő példában szereplő fájlok még *nem* léteznek.

```
(setq f (open "új.tst" "w"))      eredménye <Fájl #nnn>
(setq f (open "nincs.fil" "r"))   eredménye nil
(setq f (open "naplofj" "a"))     eredménye <Fájl #nnn>
```

A *filename* argumentum könyvtár előtagot is tartalmazhat, mint például a */teszt/funkc3* esetében. DOS operációs rendszereken a lemezmeghajtó betűjelének használata is megengedett, valamint viszahajló törtjelet (\) is használhat a törtjel (/) helyett, de ne feledkezzen meg arról, hogy egy karakterláncban a (\\) jelet kell használnia ahhoz, hogy egy visszahajló törtjelet kapjon.

```
(setq f (open "/x/uj.tst" "w"))   eredménye <Fájl #nnn>
(setq f (open "nincs.fil" "r"))   eredménye nil
```

or

Kifejezések listáján elvégzett logikai OR művelet eredményét adja vissza

(or *expr...*)

Az **or** függvény a kifejezéseket balról jobbra haladva értékeli ki, eközben egy nem-`nil` kifejezést keres. Ha talált egyet, az **or** függvény abbahagyja a további kiértékelést, és `T` értékkel tér vissza. Amennyiben mindegyik kifejezés `nil`, az **or** függvény `nil` értékkel tér vissza.

```
(or nil 45 '())          eredménye  T
(or nil '())             eredménye  nil
```

osnap

Egy háromdimenziós ponttal, mint egy adott pontra alkalmazott Tárgyraszter mód eredményével tér vissza

(osnap *pt mode*)

A *mode* argumentum egy karakterlánc, amely egy vagy több értelmezhető tárgyraszter mód azonosítóját tartalmazhatja; például `felezőpont`, `középpont`, és így tovább, egymástól vesszővel elválasztva.

```
(setq pt2 (osnap pt1 "köz"))
(setq pt3 (osnap pt1 "vég,metsz"))
```

Az **osnap** által visszaadott pont függ az aktuális háromdimenziós nézet és az APERTURE rendszerváltozó beállításától.

polar

Az FKR háromdimenziós pontját adja vissza egy ponttól megadott szöggel és távolsággal

(polar *pt angle distance*)

Az *angle* argumentum az *X* tengelytől mérve, radiánokban értendő; a szögeértékek az óramutató járásával ellentétes irányban növekednek. Bár a *pt* argumentum lehet háromdimenziós pont, az *angle* mindig az aktuális szerkesztősíkra vonatkozik.

```
(polar '(1 1 3.5) 0.785398 1.414214)  eredménye  (2.0 2.0 3.5)
```

prin1

Egy kifejezést ír a parancssorba vagy egy nyitott fájlba

(prin1 *[expr [file-desc]]*)

Az *expr* argumentumnak nem kell karakterláncnak lennie. Amennyiben a *file-desc* argumentum jelen van, és egy írásra megnyitott fájl fájlleírója, akkor a függvény az *expr* kifejezés értékét a fájlba írja, pontosan úgy, ahogyan az a képernyőn megjelenik. A függvény csak az adott *expr* kifejezést jeleníti meg, újsor vagy szóköz karaktereket nem illeszt hozzá.

```
(setq a 123 b '(a))
(prin1 'a)          kiírás A          és eredménye A
(prin1 a)           kiírás 123        és eredménye 123
(prin1 b)           kiírás (A)        és eredménye (A)
(prin1 "Helló")     kiírás "Helló"    és eredménye "Helló"
```

Az előző példák mindegyike a képernyőn jelenítette meg a kiírást, mivel nem adtuk meg a *file-desc* argumentumot. Feltételezve, hogy az `f` érvényes fájlleírója egy írásra megnyitott fájlnak, a

```
(prin1 "Helló" f)
```

a `"Helló"` karakterláncot írja a megadott fájlba, `"Helló"` értékkel tér vissza.

Ha az *expr* vezérlőkaraktereket is tartalmazó karakterlánc, a **prin1** függvény a következőképpen bővíti ki a \ jelet követő karaktereket.

Vezérlőkódok

Kód	*Jelentés*
`\\`	\ karakter
`\"`	" karakter
`\e`	Escape karakter
`\n`	Újsor karakter
`\r`	Return karakter
`\t`	Tabulátor karakter
`\`*nnn*	A nyolcas számrendszerben *nnn* kódú karakter
`\U+`*XXXX*	Unicode sorozat
`\M+N`*XXXX*	Multibájt sorozat

A következő példák mutatják a vezérlőkarakterek használatát:

```
(prin1 (chr 2))    kiírás "\002"  és eredménye  "\002"
(prin1 (chr 10))   kiírás "\n"    és eredménye  "\n"
```

A **prin1** függvény argumentumok nélkül is használható, ebben az esetben a függvény egy üres karakterklánccal tér vissza (és meg is jeleníti azt). Ha a **prin1** függvényt egy felhasználó által definiált függvényben utolsó kifejezésként használja (argumentumok nélkül), akkor a szóbanforgó függvény lefutását követően csak egy üres sor fog megjelenni, és így "csendesen" léphet ki a függvényből.

További tudnivalók

"Képernyővezérlés" című résznél, a 227. oldalon

princ

Egy kifejezést ír a parancssorba vagy egy nyitott fájlba

(princ *[expr [file-desc]]*)

Ez a függvény megegyezik a **prin1** függvénnyel, de az *expr* kifejezésben lévő vezértlőkaraktereket kibővítés nélkül jeleníti meg.

Általánosságban a **prin1** arra szolgál, hogy a **load** függvénnyel kompatibilis módon jelenítse meg a kifejezéseket, míg a **princ** olyan függvények számára szolgáltat olvasható kifejezéseket, mint a **read-line**.

print

Egy kifejezést ír a parancssorba vagy egy nyitott fájlba

(print *[expr [file-desc]]*)

Ez a függvény megegyezik a **prin1** függvénnyel, azzal a különbséggel, hogy az *expr* elé egy újsor karaktert, mögé pedig egy szóközt illeszt.

progn

Egymás után kiértékeli mindegyik kifejezést, és az utolsó kifejezés értékével tér vissza

(progn *[expr]...*)

A **progn** függvényt több kifejezés kiértékelésére használhatja olyan helyeken, ahol eredetileg csak egy kifejezés szerepelne.

```
(if (= a b)
  (progn
    (princ "\nA = B ")
    (setq a (+ a 10) b (- b 10))
  )
)
```

Az **if** függvény, amennyiben a testexpr kifejezés `nil`-től különböző értékű, normális esetben csak egyetlen *then* kifejezést értékel ki. A fenti példában viszont a **progn** függvény segítségével két kifejezést értékeltettünk ki.

prompt

Egy karakterláncot jelenít meg a képernyő prompt területén

(prompt *string*)

Kétképernyős AutoCAD konfigurációk esetében a **prompt** függvény mindkét képernyőn megjeleníti a *string* argumentumot, ezért használata előnyösebb, mint a **princ** függvényé.

`(prompt "Új érték: ")` *A képernyőn vagy képernyőkön* Új érték: *látható*

A **prompt** függvény `nil` értékkel tér vissza.

quit

Kilépteti az aktuális alkalmazást

(quit)

A **quit** függvény hívásakor a quit/exit megszakítás hibaüzenet, majd az AutoCAD Parancs promptja jelenik meg.

További tudnivalók

exit függvény

quote

A kifejezés kiértékeletlen formájával tér vissza

(quote *expr*)

Ez a következőképpen is írható:

```
'expr      ;
(quote a)                    eredménye A
(quote cat)                  eredménye CAT
(quote (a b))                eredménye (A B)
'a                           eredménye A
'cat                         eredménye CAT
'(a b)                       eredménye (A B)
```

Az utolsó három kifejezés egy AutoCAD promptra adott válaszként nem vihető be közvetlenül a billentyűzetről. Ne feledkezzen meg arról, hogy a bemenetnek egy "(" vagy "!" jellel kell kezdődnie ahhoz, hogy az AutoCAD AutoLISP kifejezésként értelmezni tudja.

read

Egy karakterlánc első listáját vagy atomját adja vissza

(read *[string]*)

A *string* argumentum nem tartalmazhat szóközöket, kivéve, ha azok egy listán vagy karakterláncon belül fordulnak elő. A **read** függvény az argumentumát a megfelelő adattípusúvá konvertálja:

```
(read "helló")              eredménye a HELLÓ atom
(read "hé te")              eredménye a HÉ atom
(read "\"Szia srácok\"")    eredménye a "Szia srácok"
                            karakterlánc
(read "(a b c)")            eredménye az (A B C) lista
(read "(a b c) (d)")        eredménye az (A B C) lista
(read "1.2300")             eredménye az 1.23 valós szám
(read "87")                 eredménye a 87 egész szám
(read "87 3.2")             eredménye a 87 egész szám
```

read-char

Egy billentyűzet bemeneti pufferéből vagy egy nyitott fájlból beolvasott karakter decimális ASCII kódját adja vissza

(read-char *[file-desc]*)

Ha nem ad meg *file-desc* argumentumot, és a billentyűzet bemeneti pufferében sincs karakter, akkor a **read-char** függvény addig vár, amíg a billentyűzet egyik billentyűjét le nem nyomja (↵ billentyűvel lezárva). Példaként tételezzük fel, hogy a billentyűzet bemeneti puffere üres. A

```
(read-char)
```

függvény arra vár, hogy Ön bebillentyűzzön valamit. Amennyiben az ABC karaktereket írja be, majd lenyomja a ↵ billentyűt, a **read-char** függvény a 65 értékkel tér vissza (ez az A betű decimális ASCII kódja).

A **read-char** következő három hívása sorra a 66, 67 és 10 (újsor) értékekkel fog visszatérni. Egy ezt követő újabb **read-char** függvényhívás már ismét bevitelre fog várni.

Különböző operációs rendszerek az ASCII szövegfájlokban eltérő módon jelölik a sor végét. A UNIX rendszerek például egyetlen újsor karaktert használnak([LF], ASCII kódja 10), míg a DOS rendszerek ugyanerre a célra egy karakterpárt ([CR]/LF, ASCII kódjuk 13 és 10) használnak. Az AutoLISP programok fejlesztésének megkönnyítése érdekében, a **read-char** függvény bármelyik jelölést elfogadja, és egyetlen újsor karakterrel tér vissza (ASCII kódja 10), amikor egy "sor vége" karaktert (vagy karaktersort) észlel.

read-line

Egy karakterláncot olvas be a billentyűzetről vagy egy nyitott fájlból

(read-line ***[file-desc]*****)**

Amennyiben a **read-line** függvény a "fájl vége" jellel találkozik, `nil` értékkel tér vissza, egyébként pedig a beolvasott karakterlánc lesz a visszaadott érték.

Példaként tételezzük fel, hogy az `f` egy érvényes nyitott fájlra mutat, ekkor a

```
(read-line f)
```

függvény a fájl következő bemeneti sorával tér vissza, illetve, ha elérte a fájl végét, akkor `nil` értékkel.

redraw

Frissíti az aktuális nézetablakot vagy egy megadott objektumot (rajzelemet) az aktuális nézetablakban

(redraw ***[ename [mode]]*****)**

A **redraw** függvény hatása a megadott argumentumoktól függ. Amennyiben a függvényt argumentumok nélkül hívja meg, akkor frissíti az aktuális nézetablakot. Ha az *ename* (rajzelemnév) argumentummal hívja meg, akkor frissíti a megadott rajzelemet. A *mode* argumentum egy egész típusú érték, amely a rajzelem frissítését szabályozza.

A redraw függvény módjai

Redraw mód	*Művelet*
1	Frissíti a rajzelemet
2	Eltünteti a rajzelemet (háttérszínnel rajzolja ki)
3	Kiemelten jeleníti meg a rajzelemet
4	A rajzelem megjelenítését visszaállítja normál módba

Ha a *name* egy összetett rajzelem (vonallánc vagy attribútumokkal rendelkező blokkreferencia) fejrészére vonatkozik, és a *mode* argumentum értéke pozitív, akkor a művelet a fő rajzelemre és valamennyi alárendelt rajzelemére is vonatkozik. Amennyiben a *mode* argumentum értéke negatív, a **redraw** függvény csak az összetett rajzelem fő elemére van kihatással.

A **redraw** függvény mindig `nil` értékkel tér vissza.

regapp

Bejegyez egy alkalmazásnevet az aktuális AutoCAD rajzba a bővített objektumadatok felhasználásának előkészítésére

(regap p *application*)

Az *application* argumentum egy maximum 31 karakter hosszú karakterlánc, melyre a szimbólumokra érvényes elnevezési szabályok vonatkoznak. Az alkalmazásnév tartalmazhat betűket, számokat, valamint speciális `$` (dollár), - (kötőjel) és _ (aláhúzás) karaktereket, de nem tartalmazhat szóközt. A névben szereplő betűket az AutoCAD nagybetűkké alakítja át.

Amennyiben az alkalmazás neve azonos egy már bejegyzett névvel, a függvény `nil` értékkel, ellenkező esetben az alkalmazás nevével tér vissza.

Amennyiben a bejegyzés sikerrel jár, az alkalmazás bekerül az APPID szimbólumtáblába. Ez a tábla a rajzban bővített adatokat használó alkalmazások listáját tartalmazza.

```
(regapp "ADESK_4153322344")
(regapp "TERVEZŐ-v2.1-124753")
```

Megjegyzés Javasoljuk, hogy egyedi alkalmazásneveket válasszon. Ennek egyik módja olyan elnevezési módszert alkalmazni, amely a vállalat vagy a termék nevéből és egy egyedi számból (például az Ön

telefonszámából vagy az aktuális dátumból/időből) áll. Az alkalmazás neve tartalmazhatja a termék verziószámát, de azt az alkalmazás egy elkülönített valós vagy egész típusú mezőben is tárolhatja; például `(1040 2.1)`.

rem

Elosztja az első számot a másodikkal, és a maradékot adja vissza

`(rem` *`number number...`*`)`

Amennyiben kettőnél több szám van megadva, a **`rem`** függvény az első számot elosztja a másodikkal és az eredményt osztja el a harmadikkal, és így tovább.

```
(rem 42 12)          eredménye 6
(rem 12.0 16)        eredménye 12.0
(rem 26 7)           eredménye 5
(rem 5 2)            eredménye 1
(rem 26 7 2)         eredménye 1
```

repeat

Annyiszor értékel ki minden egyes kifejezést, amennyi a megadott szám értéke, és az utolsó kifejezés értékével tér vissza

`(repeat` *`int expr...`*`)`

Az *`int`* argumentumnak pozitívnak kell lennie.

```
(setq a 10 b 100)
(repeat 4
  (setq a (+ a 10))
  (setq b (+ b 100))
)                    Az a 50, a b 500, a visszatérési érték 500 lesz
```

reverse

Az elemeinek sorrendjét megfordítva ad vissza egy listát

`(reverse` *`list`*`)`

```
(reverse '((a) b c))      eredménye (C B (A))
```

rtos

Egy számot karakterlánccá alakít

(rtos *number [mode [precision]]*)

Az **rtos** függvény a *number* argumentumban szereplő szám karakterláncával tér vissza, melynek formátuma a *mode*, a *precision*, valamint az AutoCAD UNITMODE rendszerváltozójának és DIMZIN méretváltozójának beállításától függ. A *mode* és *precision* argumentumok egész számok, amelyek a hosszmértékegységet és a pontosságot határozzák meg.

Hosszmértékegységek értékei

Mode értéke	*A karakterlánc formátuma*
1	Tudományos
2	Tizedes
3	Mérnöki (láb és tizedes hüvelyk)
4	Építészeti (láb és hüvelyk törtrésze)
5	Tört

A *mode* és *precision* argumentumok megfelelnek az AutoCAD LUNITS és LUPREC rendszerváltozóinak. Amennyiben elhagyja az argumentumokat, az **rtos** függvény a LUNITS és LUPREC rendszerváltozók aktuális értékét fogja alapul venni. A UNITMODE változónak csak akkor van hatása a visszatérő karakterláncra, ha mérnöki, építészeti vagy tört mértékegységet választ (a *mode* értéke 3, 4 vagy 5).

További tudnivalók

"Karakterlánc-konverziók" című résznél, a 243. oldalon, ahol folytatódik az **rtos** tárgyalása

set

Egy aposztrofált szimbólum értékét egy kifejezéshez rendeli

(set *sym expr*)

A kifejezés értékével tér vissza.

`(set 'a 5.0)` *eredménye* `5.0` *és beállítja az* `A` *szimbólum értékét*

`(set (quote b) 'a)` *eredménye* `A` *és beállítja a* `B` *szimbólum értékét*

Ha a **set** függvényt egy aposztróf nélküli szimbólumnévre alkalmazza, akkor *közvetett* módon új értéket adhat egy másik szimbólumnak. A fenti példából kiindulva

`(set b 640)` *eredménye* `640`

és a 640 értéket az `a` szimbólumhoz rendeli hozzá, mivel `b` szimbólum ezt tartalmazza.

További tudnivalók

setq függvény

set_tile

Egy párbeszédpanel-zóna értékét állítja be

(set_tile *key value*)

A *key* argumentum a zónát, a *value* pedig a hozzárendelni kívánt új értéket adja meg (kezdeti értékét a `value` attribútum állította be). Mindkét argumentum karakterlánc típusú.

setcfg

Alkalmazási adatokat ír az *acad.cfg* fájl AppData szakaszába

(setcfg *cfgname cfgval*)

A *cfgname* argumentum egy karakterlánc, amely megadja a szakaszt és azt az értéket, amelyre a *cfgval* argumentumot kell állítani. Amennyiben a *cfgname* nem érvényes, a **setcfg** `nil` értékkel tér vissza. A *cfgname* argumentumnak a következő formátumú karakterláncnak kell lennie :

`"AppData/alkalmazásnév/szakasznév/.../param_név"`

A következő programrészlet az AppData/ArchStuff szakaszban lévő WallThk paraméter értékét 8-ra állítja és a "8" karakterlánccal tér vissza:

```
(setcfg "AppData/ArchStuff/WallThk" "8")   eredménye "8"
```

További tudnivalók

getcfg függvény

setfunhelp

Egy felhasználó által definiált parancsot jegyez be a Súgóba, azért, hogy a megfelelő súgófájl és téma kerüljön meghívásra, amikor a felhasználó segítséget kér az adott paranccsal kapcsolatban

(setfunhelp *c:fname [helpfile [topic [command]]]*)

A *c:fname* argumentum egy karakterlánc, amely megadja a felhasználó által definiált parancsot (a **C:XXX** függvényt) és amelynek tartalmaznia kell a **C:** előtagot. A **defun** függvénynek átadott *function* argumentummal ellentétben a **setfunhelp** függvény argumentumát idézőjelbe kell tenni. A fennmaradó három opcionális argumentum a végrehajtandó súgóhívást írja le. Ezek szintén karakterláncok, és megegyeznek a **help** függvénynek átadott argumentumokkal. Sikeres hívás esetén a **setfunhelp** függvény a *c:fname* argumentumnak átadott karakterlánccal tér vissza, egyébként nil a visszatérési érték.

Meg kell jegyezni, hogy ez a függvény csak azt ellenőrzi, hogy a *c:fname* argumentumnak megvan-e a C: előtagja. *Nem* ellenőrzi viszont a *c:fname* függvény meglétét és a további argumentumok helyességét.

Amikor Ön a **defun** függvényt használja egy **C:XXX** függvény definiálására, akkor az eltávolítja a függvény nevét a **setfunhelp** által bejegyzett függvények nevei közül (amennyiben létezik ilyen). Ezért a **setfunhelp** függvényt a **defun** hívás után kell meghívnia, amely meghatározza a felhasználó által definiált parancsot.

A következő példa a **defun** függvényt használja a felhasználó által definiált SAJÁTFÜGGV parancs megadására. A **setfunhelp** függvény bejegyzi a **C:SAJÁTFÜGGV** nevű függvényt és azt a sajátfüggv témával azonosítja az *ensugom.ahp* fájlban.

```
(defun c:sajátfüggv ()
  ...
  (getint "adjál: ")
  ...
)
(setfunhelp "c:sajátfüggv" "ensugom.ahp" "sajátfüggv")
```

Parancs: **sajátfüggv**
adjál: **'súgó**

Feltéve, hogy az *ensugom.ahp* AutoCAD Súgófájl létezik a "support" elérési útvonalon, az AutoCAD megjeleníti a Súgó párbeszédet a *sajatalk.ahp* fájlból vett sajátfüggv témával.

További tudnivalók

defun és **help** függvény

setq

Egy vagy több szimbólum értékét társított kifejezésekhez rendeli

(setq *sym1 expr1 [sym2 expr2]...*)

Ez az AutoLISP alapvető értékadó függvénye. Egyetlen függvényhívással több szimbólumnak is adhat értéket, de a függvény csak az utolsó *expr* kifejezés értékével tér vissza.

```
(setq a 5.0)                    eredménye  5.0
```

és az a szimbólumhoz az 5.0 értéket rendeli hozzá. Ezt követően az a kiértékelésekor mindig az 5.0 valós számot értékeli ki.

```
(setq b 123 c 4.7)              eredménye  4.7
(setq s "az")                   eredménye  "az"
(setq x '(a b))                 eredménye  (A B)
```

További tudnivalók

"AutoLISP változók" című résznél, a 196. oldalon

setvar

Egy AutoCAD rendszerváltozót megadott értékre állít

(setvar *varname value*)

Sikeres hívás esetén a **setvar** függvény a rendszerváltozó értékével tér vissza. A rendszerváltozó nevét idézőjelek közé kell tenni.

```
(setvar "FILLETRAD" 0.50)  eredménye  0.5
```

és AutoCAD lekerekítési sugarát 0.5 egységre állítja be. Egész típusú értékkel rendelkező rendszerváltozók esetében a megadott *value* argumentumnak –32768 és +32767 közé kell esnie.

Bizonyos AutoCAD parancsok, még mielőtt bármilyen promptot kiadnának, a rendszerváltozóktól vesznek át értékeket. Ha a **setvar** függvény segítségével a folyamatban lévő parancs használata során ad új értéket ezen rendszerváltozóknak, akkor az újonnan beállított érték csak a következő AutoCAD parancs használatakor fejti ki hatását.

Megjegyzés Amikor a **setvar** függvényt az AutoCAD ANGBASE rendszerváltozójának beállítására használja, az érték radiánban értendő. Ez különbözik az AutoCAD VÁLTOZÓK parancsától, mely ezt az argumentumot fokokban értelmezi. Abban az esetben, ha a SNAPANG rendszerváltozó beállítására használja a **setvar** függvényt, akkor az érték radiánban értendő az alapértelmezés szerinti 0 szöghöz viszonyítva, amely *keleti* iránynak, vagy *3 órának* felel meg. Ez szintén különbözik a VÁLTOZÓK parancstól, amely ezt az argumentumot fokokban értelmezi az ANGBASE rendszerváltozó beállításához képest.

Megjegyzés A VISSZA parancs nem érvényteleníti a CVPORT rendszerváltozó értékének a **setvar** függvény által történt megváltozását.

Az AutoCAD jelenlegi rendszerváltozóinak felsorolása az AutoCAD on-line súgójában található .

További tudnivalók

getvar függvény

sin

Egy szög szinuszát adja eredményül radiánban kifejezett valós számként

(sin *angle*)

Az *angle* argumentumot radiánban kell megadni .

```
(sin 1.0)          eredménye  0.841471
(sin 0.0)          eredménye  0.0
```

slide_image

Egy AutoCAD diát rajzol az éppen aktív párbeszédpanel ábrazónájába

(slide_image *x1 y1 wid hgt sldname*)

A dia lehet diafájl (*.sld*), vagy diakönyvtár (*.slb*): a `sldname` argumentum olyan alakban adja meg, ahogyan azt Ön tenné egy BEDIA parancs vagy egy menüfájl esetében (további információ az "Ábrák létrehozása" című résznél, a 320. oldalon található):

`sldname` *vagy* `libname(sldname)`

A dia első (bal felső) sarokpontja —beillesztési pontja— az (`x1,y1`)-ben; átellenes (jobb alsó) sarokpontja az első ponttól (`wid,hgt`) relatív távolságra helyezkedik el (a `wid` és a `hgt` argumentumoknak pozitív értékeknek kell lennie). Az origó (0,0) az ábra bal felső sarka. A jobb alsó sarok koordinátáit a méretfügvények (**a dimx_tile** és a **dimy_tile**) hívásával kaphatja meg.

snvalid

Ellenőrzi a szimbólumtábla-nevet az érvényes karakterek szempontjából

(snvalid *sym_name [flag]*)

A `sym_name` argumentum egy karakterlánc, amely egy szimbólumtábla-nevet ad meg. Az opcionális `flag` argumentum értéke 1 vagy 0 lehet (alapértelmezés szerint 0). Az **snvalid** függvény `T` értékkel tér vissza, ha a `sym_name` érvényes, egyébként `nil` a visszatérési érték.

A szimbólumtábla-nevek csak alfanumerikus karakterekből és a $(dollár), _(aláhúzás) és a – (kötőjel) speciális karakterekből állhatnak. Az üres karakterlánc nem érvényes név.

sqrt

Egy szám négyzetgyökét adja vissza valós számként

(sqrt *num*)

```
(sqrt 4)          eredménye 2.0
(sqrt 2.0)        eredménye 1.41421
```

ssadd

Hozzáad egy objektumot (rajzelemet) egy kiválasztási halmazhoz, vagy új kiválasztási halmazt hoz létre

(ssadd *[ename [ss]]*)

Az **ssadd** függvény új, elemek nélküli kiválasztási halmazt hoz létre, ha argumentumok nélkül kerül meghívásra. Ha egyetlen *ename* rajzelemnév-argumentummal hívja meg, akkor az **ssadd** függvény olyan kiválasztási halmazt hoz létre, amely ezt az egy elemet tartalmazza. Amennyiben egy elemnévvel és egy *ss* kiválasztási halmazzal hívja meg, akkor a függvény a megnevezett rajzelemet hozzáadja a kiválasztási halmazhoz. Az **ssadd** függvény mindig az új vagy módosított kiválasztási halmazzal tér vissza.

Amennyiben egy rajzelemet hozzáad egy halmazhoz, ez a rajzelem hozzáadódik a létező halmazhoz, és a függvény az így kapott, *ss* argumentummal megadott halmazzal tér vissza. Ez azt jelenti, hogy ha a halmaz más változókhoz is hozzá van rendelve, akkor azok tartalma szintén tükrözni fogja a változást. Amennyiben a nevezett rajzelem már szerepel a halmazban, a rendszer az **ssadd** műveletről nem vesz tudomást, és hibajelzést sem ad.

```
(setq e1 (entnext))          Az e1 értéke a rajz első rajzelemének neve lesz
(setq ss (ssadd))            Létrehozza az ss nevű, üres kiválasztási halmazt
(ssadd e1 ss)                Az ss halmazhoz hozzáadja az e1 elemet
(setq e2 (entnext e1))       Az e1 elemet követő rajzelemet adja meg
(ssadd e2 ss)                Az e2 elemet hozzádaja az ss-hez, és ss-sel tér vissza
```

ssdel

Kitöröl egy objektumot (rajzelemet) egy kiválasztási halmazból

(ssdel *ename ss*)

Az **ssdel** függvény kitörli az *ename* nevű elemet az *ss* nevű kiválasztási halmazból, és az *ss* nevével tér vissza. Meg kell jegyezni, hogy a függvény nem egyszerűen egy, a megadott elem nélküli új halmazzal tér vissza, hanem a rajzelem ténylegesen is kitörlődik az adott kiválasztási halmazból. Ha a rajzelem nem szerepel a halmazban, akkor a függvény `nil` értékkel tér vissza.

Például, ha az `e1` nevű elem szerepel az `ss1` halmazban, az `e2` pedig nem, akkor

```
(ssdel e1 ss1)     eredménye az e1 nélküli ss1 kiválasztási halmaz
(ssdel e2 ss1)     eredménye nil (az ss1 változatlan marad)
```

ssget

Objektumok (rajzelemek) kiválasztására kéri a felhasználót, és egy kiválasztási halmazzal tér vissza

(ssget *[mode] [pt1 [pt2]] [pt-list] [filter-list]*)

A *mode* argumentum egy karakterlánc, amely az alkalmazott rajzelem-kiválasztási módszert határozza meg. Az érvényes módok az AutoCAD Ablak, APoligon, Metsz, MPoligon, Utolsó, Előző, előVálasztott vagy Felfűz kiválasztási módjainak megfelelően `"A"`, `"AP"`, `"M"`, `"MP"`, `"U"`, `"E"`, `"V"` és `"F"` lehetnek. A *mode* értékének egy további opciója az `"X"`, amely a teljes adatbázist kiválasztja. A *pt1* és *pt2* argumentumok a kiválasztáshoz felhasználni kívánt pontokat határozzák meg. Amennyiben a *mode* argumentum nélkül ad meg egy pontot, ez a pontkijelöléssel történő rajzelem-kiválasztásnak felel meg. A Traszter aktuális beállítását a függvény figyelmen kívül hagyja, hacsak azt Ön a függvény futása közben kifejezetten nem kéri. A *filter-list* (szűrőlista) argumentum egy asszociációs lista, amely megadja az objektum tulajdonságait. A függvény a *filter-list* argumentumnak megfelelő objektumokat hozzáadja a kiválasztási halmazhoz. Amennyiben az összes argumentum elmarad, akkor az **ssget** függvény a Válasszon objektumokat prompt megjelenítésével kéri a felhasználót, hogy interaktív módon állítson össze egy kiválasztási halmazt.

A kiválasztási halmaz egyaránt tartalmazhat papír- és modelltérbeli rajzelemeket, da amikor a kiválasztási halmazt felhasználja egy műveletben, akkor az éppen nem használt tér elemeit az AutoCAD automatikusan kiszűri. Az **ssget** függvénnyel kiválasztott halmaz csak fő rajzelemeket tartalmazhat (attribútumokat vagy vonalláncok csomópontjait nem).

```
(ssget)
```
A felhasználótól általános rajzelem-kiválasztást kér, és a rajzelemeket egy kiválasztási halmazban helyezi el

```
(ssget "E")
```
A legutóbb kiválasztott objektumokból képez kiválasztási halmazt

```
(ssget "U")
```
Az adatbázishoz utoljára hozzáadott, látható rajzelemből képez kiválasztási halmazt

```
(ssget "V")
```
Az előválasztott kiválasztási halmazban lévő lévő rajzelemekből (melyek kiválasztása a PICKFIRST aktív állapotában történt) képez kiválasztási halmazt

```
(ssget '(2 2))
```
A (2,2) ponton átmenő rajzelemből képez kiválasztási halmazt

```
(ssget "A" '(0 0) '(5 5))
```
A (0,0) és (5,5) pontok által kifeszített ablakon belüli rajzelemekből képez kiválasztási halmazt

`(ssget "M" '(0 0) '(1 1))`	*A (0,0) és (1,1) pontok által kifeszített ablakkal elmetszett rajzelemekből képez kiválasztási halmazt*
`(ssget "X")`	*Az adatbázis összes rajzeléméből képez kiválasztási halmazt*
`(ssget "X" filter-list)`	*Végigpásztázza az adatbázist, és a szűrőlistának megfelelő rajzelemekből képez kiválasztási halmazt*
`(ssget filter-list)`	*A felhasználótól általános rajzelem-kiválasztást kér, de csak a szűrőlistának megfelelő rajzelemeket helyezi el a kiválasztási halmazban*
`(ssget "P" filter-list)`	*A legutóbb kiválasztottak közül azokból a rajzelemekből képez kiválasztási halmazt, amelyek megfelelnek a szűrőlistának*

Az alábbi példákban az **ssget** függvény egy pontokat tartalmazó lista átadását igényli.

```
(setq pt_list '((1 1)(3 1)(5 2)(2 4)) )
```

`(ssget "AP" pt_list)`	*A* `pt_list` *által meghatározott poligon belsejében található összes rajzelemből képez kiválasztási halmazt*
`(ssget "MP" pt_list)`	*A* `pt_list` *által meghatározott poligon belsejében található, valamint azzal elmetszett összes rajzelemből képez kiválasztási halmazt*
`(ssget "F" pt_list)`	*A* `pt_list` *által meghatározott, felfűző vonalak által elmetszett összes rajzelemből képez kiválasztási halmazt*
`(ssget "AP" pt_list filter-list)`	*A* `pt_list` *által meghatározott poligon belsejében található összes rajzelem közül azokból képez kiválasztási halmazt, amelyek megfelelnek a szűrőlistának*

A kiválasztott objektumok a képernyőn csak akkor kerülnek kiemelésre, ha az **ssget** függvényt argumentumok nélkül használja. A kiválasztási halmazok az AutoCAD ideiglenes fájljaitól vesznek el helyet, ezért az AutoLISP 128-nál több halmazt nem tarthat egyszerre nyitva. Amennyiben ezt a határt eléri, az AutoCAD megtagadja a további kiválasztási halmazok létrehozását, és minden **ssget** függvényhívás `nil` értékkel tér vissza. Egy már szükségtelen kiválasztási halmazt úgy zárhat le, ha az azt tartalmazó változónak `nil` értéket ad.

Egy kiválasztási halmazt tartalmazó változóval az AutoCAD minden olyan Válasszon objektumokat promptjára válaszolhat, amely esetében az Utolsó kiválasztási módszer megengedett. Ez kiválasztja a változóhoz rendelt halmazban lévő összes objektumot.

Kiválasztási halmazok szűrői

A halmazok kiválasztására szolgáló szűrőlisták bármely kiválasztási móddal együtt használhatók. Ezen módszerek segítségével olyan kiválasztási halmazhoz juthat, amely az adott típusú, adott fólián elhelyezkedő vagy adott színű összes objektumot tartalmazza.

A következő példa eredménye egy olyan kiválasztási halmaz, amely kizárólag az előválasztott (a PICKFIRST aktív állapotában kiválasztott rajzelemeket tartalmazó) halmaz részét képező, kék színű vonalakat foglalja magában:

```
(ssget "I" '((0 . "LINE") (62 . 5)))
```

Az **ssget** szűrőlistájának segítségével kiválaszthatja az összes olyan rajzelemet, amely egy adott alkalmazáshoz tartozó bővített rajzelem-adatot tartalmaz. Ezt a –3-as csoportkód felhasználásával teheti meg, például így:

```
(ssget "V" '((0 . "CIRCLE") (-3 ("ALKNÉV"))))
```

Ezzel kiválasztja az összes olyan kört, amely az `"ALKNÉV"` alkalmazáshoz kapcsolódó adatot tartalmaz.

További információ a "Kiválasztási halmazok szűrőlistái" című résznél, a 266. oldalon és "Bővített adatok szűrése" című résznél, a 269. oldalon található.

Relációvizsgálat

Egyéb rendelkezés hiányában a *`filter-list`* elemeit a függvény "egyenlőségre" vizsgálja. Számszerű értékeket tartalmazó csoportok (egészek, valós számok, pontok és vektorok) esetében egyéb relációk is megadhatók egy relációjelet meghatározó speciális –4-es csoportkód beiktatásával. A –4-es csoport értéke egy karakterlánc, amely a szűrőlista következő csoportjának vizsgálatára vonatkozó relációjelet tartalmazza. Pédául:

```
(ssget "X" '((0 . "CIRCLE") (-4 . ">=") (40 . 2.0)))
```

minden olyan kört kiválaszt, amelynek sugara (csoportkódja 40) nagyobb vagy egyenlő, mint 2.0 egység.

A következő táblázat a lehetséges relációjeleket mutatja be:

Relációjelek a halmazok kiválasztására szolgáló szűrőlistákban

Relációjel	*Jelentés*
`"*"`	Bármi lehet (mindenképpen igaz)
`"="`	Egyenlő

Relációjelek a halmazok kiválasztására szolgáló szűrőlistákban (*folytatás*)

Relációjel	*Jelentés*
"!="	Nem egyenlő
"/="	
"<>"	
"<"	Kisebb
"<="	Kisebb vagy egyenlő
">"	Nagyobb
">="	Nagyobb vagy egyenlő
"&"	Bitenkénti AND (csak egész típusú csoportok esetében)
"&="	Bitenként maszkolt egyenlőség (csak egész típusú csoportok esetében)

A relációjelek használata a vizsgálni kívánt csoportok fajtájától függ:

- A valós vagy egész számértékeket tartalmazó csoportok esetében valamennyi relációjel érvényes, kivéve a bitenkénti relációk jeleit ("&" és "&=").
- A "&" és "&=" bitenkénti relációk jelei csak egész típusú csoportok esetében használhatók. A bitenkénti AND ("&") akkor ad igaz értéket, ha (*(egész_csoport & szűrő)* /= 0), azaz ha a maszkban beállított bitek bármelyike az egész típusú csoportban is be van állítva. A bitenként maszkolt egyenlőség ("&=") akkor teljesül, ha (*(egész_csoport & szűrő)* = *szűrő*), azaz ha a maszkban beállított valamennyi bit be van állítva az *egész_csoport*-ban is (az *egész_csoport* egyéb bitjei is be lehetnek állítva, a vizsgálat azonban ezekre nem terjed ki).
- Pont típusú csoportok esetében az *X*, *Y* és *Z* koordináták vizsgálata egyetlen karakterlánccal végezhető, amelyben az egyes relációjeleket vesszők választják el egymástól (például ">,>,*"). Ha a karakterláncból hiányzik az egyik relációjel (pédául, "=,<>", kihagyja a *Z* koordinátát a vizsgálatból), akkor azt úgy kell értelmezni, mintha a "bármi lehet" jel, a "*" állna azon a helyen.
- Az irányvektorok (210-es csoport) összehasonlítása csak a "*", "=" és "!=" (valamint ez utóbbival egyenértékű "nem egyenlő") relációjelekkel történhet.
- A relációjelek nem használhatók a karakterlánc típusú csoportok esetében; vizsgálatukra a globális karakterek alkalmasak.

Szűrővizsgálatok logikai csoportosítása

A fentiekben ismertetett relációjelek bináris műveleteket jelölnek. A csoportok vizsgálatára az alábbi táblázatban látható csoportosító operátorok alkalmazásával beágyazott logikai (Boolean) kifejezéseket is létrehozhat. A csoportosító operátorokat a relációjelekhez hasonlóan –4-es csoportkód segítségével adhatja meg. Párosával fordulnak elő, és a párosításnak a szűrőlistában tökéletesen kiegyenlítettnek kell lennie, máskülönben az **ssget** függvény hívása sikertelen lesz. Az operátorok által közrezárható operandusok száma az adott művelettől függ.

Csoportosító operátorok a halmazok kiválasztására szolgáló szűrőlistákban

Nyitó operátor	*Közrezárt operandusok*	*Záró operátor*
"<AND"	Egy vagy több operandus	"AND>"
"<OR"	Egy vagy több operandus	"OR>"
"<XOR"	Két operandus	"XOR>"
"<NOT"	Egy operandus	"NOT>"

A csoportosító operátorok esetében az *operandus* egy rajzelem-mezőt tartalmazó csoport, egy relációjel és egy azt követő egy rajzelem-mezőt tartalmazó csoport, vagy egy ezen operátorok által létrehozott beágyazott kifejezés lehet. A következő példa a csoportosító operátorok használatát mutatja be egy szűrőlistában:

```
(ssget "X" '((-4 . "<OR")
             (-4 . "<AND")
               (0 . "CIRCLE")
               (40 . 1.0)
             (-4 . "AND>")
             (-4 . "<AND")
               (0 . "LINE")
               (8 . "ABC")
             (-4 . "AND>")
           (-4 . "OR>"))
)
```

Ez a függvényhívás kiválasztja mindazokat a köröket, amelyek sugara 1.0 egység, valamint mindazokat a vonalakat, amelyek az `"ABC"` fólián helyezkednek el.

A csoportosító operátorok nagy- és kisbetűs írásmódja között nincs különbség, így használhatja a kisbetűs megfelelőjüket is: `"<and"`, `"and>"`, `"<or"`, `"or>"`, `"<xor"`, `"xor>"`, `"<not"` és `"not>"`.

sslength

Egy kiválasztási halmazban lévő objektumok (rajzelemek) számát adja meg

(sslength *ss*)

Amennyiben a szám nagyobb mint 32767, valós típusú lesz. A kiválasztási halmazban soha nem szerepel kétszer ugyanaz az elem.

```
(setq sset (ssget "U"))      Az utolsó objektumot elhelyezi
                             az sset kiválasztási halmazban
(sslength sset)              eredménye 1
```

ssmemb

Megvizsgálja, hogy egy objektum (rajzelem) szerepel-e egy kiválasztási halmazban

(ssmemb *ename ss*)

Ha igen, akkor az **ssmemb** rajzelem nevével (*ename*), ha nem, akkor `nil` értékkel tér vissza. Például, ha az `e1` nevű rajzelem szerepel az `ss1` a kiválasztási halmazban, az `e2` nevű viszont nem, akkor

```
(ssmemb e1 ss1)              eredménye az e1 elemnév
(ssmemb e2 ss1)              eredménye nil
```

ssname

Egy kiválasztási halmaz megadott sorszámú (indexű) elemének objektum (rajzelem) nevét adja vissza

(ssname *ss index*)

Az *index* argumentumnak egész típusúnak kell lennie. Amennyiben az *index* értéke negatív, vagy nagyobb, mint a kiválasztási halmaz legmagasabb indexű elemének indexe, a függvény `nil` értékkel tér vissza. A halmaz első elemének indexe 0. Az **ssget** függvénnyel képzett kiválasztási halmazokban csak fő rajzelemek nevei találhatók. Alárendelt rajzelemekhez (attribútumok és vonallánc-csomópontok) ezzel a módszerrel nem férhet hozzá. (Ezek elérésére az **entnext**

függvény nyújt lehetőséget. További információ az **"entnext"** függvénynél található)

```
(setq sset (ssget))                    Létrehoz egy sset nevű kiválasztási halmazt
(setq ent1 (ssname sset 0))            Kiolvassa az első rajzelem nevét az sset-ből
(setq ent4 (ssname sset 3))            Kiolvassa a negyedik rajzelem nevét az sset-ből
```

A 32767-nél magasabb indexű rajzelemek eléréséhez az *index* argumentumot valós számként kell megadni. Például:

```
(setq entx (ssname sset 50843.0))      Kiolvassa az sset 50844. elemének nevét
```

startapp

Elindít egy Windows alkalmazást

(startapp *appcmd file*)

Az *appcmd* argumentum egy karakterlánc, amely a végrehajtani kívánt alkalmazást adja meg. Amennyiben az *appcmd* nem tartalmazza a teljes elérési útvonalat, a függvény a PATH környezeti változóban található útvonalon keresi az alkalmazást. A *file* argumentum egy karakterlánc, amely a megnyitni kívánt fájl nevét tartalmazza. Sikeres hívás esetén egy 0-nál nagyobb egésszel tér vissza, egyébként pedig 0-val.

A következő kód a Windows Notepad alkalmazását indítja el, és megnyitja az *acad.lsp* fájlt.

```
(startapp "notepad" "acad.lsp")
```

Külsőleg definiált függvény

Csak Windows, *acadapp* ADS alkalmazás

start_dialog

Megjelenít egy párbeszédpanelt, és megindítja a felhasználói bevitel fogadását

(start_dialog)

A párbeszédpanelt ezt megelőzően egy **new_dialog** hívással inicializálni kell. A párbeszédpanel mindaddig aktív marad, amíg egy akciókifejezés vagy visszajelzési függvény nem hívja meg a **done_dialog** függvényt. A **done_dialog** rendszerint ahhoz a zónához

kötődik, amelynek `key` attribútuma `"accept"` (jellemzően az OK nyomógomb) vagy `"cancel"` (jellemzően Mégsem nyomógomb).

A **`start_dialog`** függvénynek nincs argumentuma. Visszatérési értéke a **`done_dialog`** hívásnak opcionálisan átadott *`status`*. Ennek alapértelmezés szerinti értéke `1`, ha a felhasználó az OK, és `0`, ha a felhasználó a Mégsem nyomógombot nyomta meg; illetve `-1`, ha egy **`term_dialog`** hívás az összes párbeszédpanelt lezárta. Amennyiben azonban a **`done_dialog`** hívás *`status`* argumentumaként egy 1-nél nagyobb számot ad meg, akkor a **`start_dialog`** ezzel az értékkel tér vissza, amelynek jelentése az alkalmazástól függ.

start_image

Megindítja egy ábra szerkesztését a párbeszédpanel zónájában

`(start_image` ***`key`***`)`

Az ezt követő **`fill_image`**, **`slide_image`** és **`vector_image`** hívásoknak addig van hatásuk az ábrára, amíg az alkalmazás az **`end_image`** függvényt meg nem hívja. A *`key`* argumentum egy karakterlánc, amely megadja a párbeszédpanel zónáját. A *`key`* argumentum különbséget tesz a kis- és nagybetűk között.

start_list

Megindítja a listaablak vagy legördülő lista párbeszédpanel zónájában található lista feldolgozását

`(start_list` ***`key [operation [index]]`***`)`

A *`key`* argumentum egy karakterlánc, amely megadja a párbeszédpanel zónáját. A *`key`* argumentum megkülönbözteti a kis- és nagybetűket. Az *`operation`* argumentum egy egész szám, amelynek jelentését a következő táblázatban foglaltuk össze .

Listapanelkódok a start_list függvényhez

Érték	*Leírás*
1	A kiválasztott lista tartalmának megváltoztatása
2	Új tétel hozzáfűzése a listához
3	A régi lista törlése és új lista létrehozása (ez az alapértelmezés)

Az *`index`* argumentumot a függvény csak akkor veszi figyelembe, ha a **`start_list`** hívás a lista megváltoztatására irányul (1), amely esetben

az *index* az ezt követő **add_list** hívással megváltoztatni kívánt tételt jelöli. Az első *index* a 0. Amennyiben nem adja meg az *operation* argumentumot, akkor annak alapértelmezés szerinti értéke 3 (új lista létrehozása), ha pedig az *operation* argumentumot megadja, de az *index* argumentumot nem, akkor az alapértelmezés szerinti index a 0.

Az egymás után meghívott **add_list** függvények mindaddig hatással vannak a **start_list** által indított listára, amíg az alkalmazás meg nem hívja az **end_list** függvényt.

strcase

A karakterlánc nagy- vagy kisbetűre konvertált változatát adja vissza

(strcase *string [which]*)

Amennyiben a *which* argumentum nincs megadva, vagy értéke `nil`, a *string* argumentum minden betűjét nagybetűvé konvertálja. Ha a *which* argumentum szerepel a függvényhívásban, és értéke nem `nil`, akkor az összes betű kisbetű lesz.

```
(strcase "Minta")        eredménye "MINTA"
(strcase "Minta" T)      eredménye "minta"
```

A **strcase** függvény az aktuálisan konfigurált karakterkészletnek megfelelően rendeli egymáshoz a kis- és nagybetűket (további információ a "Nemzeti (honosított) nyelv-támogatás" című részné, a 225. oldalon található).

strcat

Egy karakterlánccal tér vissza, amely több karakterlánc összefűzéséből jött létre

(strcat *string1 [string2]...*)

```
(strcat "ar" "ról")      eredménye "arról"
(strcat "a" "b" "c")     eredménye "abc"
(strcat "a" "" "c")      eredménye "ac"
```

strlen

Egy karakterláncban található karakterek számával tér vissza

(strlen *[string]...*)

Amennyiben több *string* argumentum van megadva, akkor az összes argumentum hosszának összegével tér vissza. Az argumentumok elhagyásának vagy egy üres karakterlánc magadásának eredménye 0.

```
(strlen "abcd")                  eredménye  4
(strlen "ab")                    eredménye  2
(strlen "egy" "hat" "négy")eredménye  10
(strlen)                         eredménye  0
(strlen "")                      eredménye  0
```

subst

Másolatot ad vissza egy listáról, ahol a régi tétel minden előfordulása helyén az új tétel szerepel

(subst *newitem olditem list*)

Amennyiben az *olditem* argumentumot nem találja a listában, a **subst** függvény a *list* argumentumot változtatás nélkül adja vissza.

```
(setq sample '(a b (c d) b))
(subst 'qq 'b minta)             eredménye  (A QQ (C D) QQ)
(subst 'qq 'z minta)             eredménye  (A B (C D) B)
(subst 'qq '(c d) minta)         eredménye  (A B QQ B)
(subst '(qq rr) '(c d) minta)    eredménye  (A B (QQ RR) B)
(subst '(qq rr) 'z minta)        eredménye  (A B (C D) B)
```

A **subst** függvényt az **assoc** függvénnyel együtt használva, egy asszociációs lista adott kulcshoz tartozó értékeit egyszerűen lecserélheti.

```
(setq kicsoda '((első jani) (közép q) (hátsó emb)))

(setq régi (assoc 'első kicsoda)  eredménye  (ELSŐ JANI)
      új   '(első j)              eredménye  (ELSŐ J)
)
(subst új régi kicsoda)           eredménye
                                    ((ELSŐ J)(KÖZÉP Q)(HÁTSÓ EMB))
```

substr

Egy karakterlánc részkarakterláncát adja vissza eredményül

(substr *string start [length]*)

A **substr** függvény a *start* argumentum által megadott karakterpozíciótól induló és a *length* argumentum által megadott

számú karaktert tartalmazó karakterláncccal tér vissza. Amennyiben a *length* argumentum nincs megadva, a részkarakterlánc a *string* végéig fog tartani. A *start* és a *length* argumentumok értékeinek pozitív egész számoknak kell lenniük.

A *string* első karaktere az 1-es számú karakter. Ez különbözik a listaelemekkel foglalkozó összes többi függvénytől (mint az **nth** és **ssname**) , amelyek az első elemet a 0 sorszámmal kezelik.

```
(substr "abcde" 2)        eredménye "bcde"
(substr "abcde" 2 1)      eredménye "b"
(substr "abcde" 3 2)      eredménye "cd"
```

tablet

Lekérdezi és beállítja a digitalizáló eszköz (tábla) kalibrációját

(tablet *code [row1 row2 row3 direction]*)

A *code* argumentumban megadott egész számtól függően, a **tablet** függvény vagy lekérdezi a digitalizáló eszköz aktuális kalibrációját, vagy beállítja azt. Amennyiben a *code* értéke 0, a **tablet** függvény az aktuális kalibrációval tér vissza. Ha a *code* értéke 1, akkor azt új beállítás értékeinek kell követniük : *row1, row2, row3* és *direction*.

code Egész szám. Amennyiben a *code* értéke 0, a **tablet** függvény az aktuális kalibrációval tér vissza; ebben az esetben a többi argumentumot el kell hagyni. Ha a megadott kód értéke 1, a **tablet** függvény a többi argumentumnak megfelelően állítja be a kalibrációt; ebben az esetben meg kell adni a többi argumentumot is.

row1, row2, row3 Három térbeli (háromdimenziós) pont. Ezek az argumentumok határozzák meg a tábla transzformációs mátrixának három sorát.

direction Egy térbeli (háromdimenziós) pont. Ez a tábla felülete által képviselt síkra merőleges normálvektor (a VKR-ben, a Világ koordináta-rendszerben kifejezve) .

Megjegyzés Amennyiben a megadott *direction* argumentum nem normalizált, a **tablet** függvény helyesbíti, így a kalibráció beállításakor a függvény által visszaadott *direction* (irány) különbözhet az argumentumban megadott értéktől. Hasonlóképpen, a *row3* harmadik elemének (*Z*) mindig 1-gyel kell egyenlőnek lennie: a **tablet** függvény mindig 1-et ad vissza ezen a helyen, még ha a *row3* lista más értéket tartalmaz is.

Hiba esetén a **tablet** függvény `nil` értékkel tér vissza, és az ERRNO rendszerváltozóban elhelyezi a hiba okát jelző értéket (további információ a 17., "AutoLISP hibakódok és hibaüzenetek" című

fejezetben található). Ez történik például abban az esetben, ha nem tábla a digitalizáló eszköz.

A **tablet** függvénnyel meghatározható legegyszerűbb transzformáció az azonosság:

```
(tablet 1 '(1 0 0) '(0 1 0) '(0 0 1) '(0 0 1))
```

Amennyiben ez a transzformáció van érvényben, az AutoCAD a digitalizáló tábla tényleges, feldolgozatlan koordinátáit fogja megkapni. Például, amennyiben a digitalizáló (5000,15000) koordinátájú pontját jelöli ki, akkor azt az AutoCAD az aktuális rajz ugyanilyen koordinátájú pontjának fogja tekinteni .

A TABMODE rendszerváltozó segítségével az AutoLISP rutinok be- vagy ki tudják kapcsolni a tábla módot.

További tudnivalók

"A digitalizáló táblák kalibrálása" című résznél, a 257. oldalon további információ a tábla transzformációs mátrixszal kapcsolatban

tblnext

Megkeresi egy szimbólumtábla következő tételét

(tblnext *table-name [rewind]*)

A *table-name* argumentum egy karakterlánc, amely a szimbólumtáblát azonosítja. Lehetséges *table-name* értékek "LAYER", "LTYPE", "VIEW", "STYLE", "BLOCK", "UCS", "APPID", "DIMSTYLE", és "VPORT". A karakterláncnak nem szükséges nagybetűkből állnia.

Megjegyzés Mivel a **vports** függvény visszaadja az aktuális VPORT tábla információit, egyszerűbb, ha a **vports** függvényt használja ennek az információnak a lekérdezésére a **tblnext** függvény helyett.

Ha a **tblnext** függvényt egymás után többször használja, akkor az normális esetben mindig a megadott tábla következő elemével tér vissza. A **tblsearch** függvénnyel beállítható a *következő* lekérdezni kívánt táblabejegyzést. Ha azonban a *rewind* argumentumot is megadta, és értéke nem nil, akkor a függvény visszatér a függvénytábla elejére, és annak első bejegyzését olvassa ki. Amennyiben nincs több bejegyzés a táblában, a visszatérési érték nil. A függvény törölt táblabejegyzéssel soha nem tér vissza.

Amennyiben megtalálta a keresett bejegyzést, a függvény egy DXF típusú kódokat és értékeket tartalmazó kapcsolt értékpárból álló listával tér vissza.

```
(tblnext "layer" T)                    Lekérdezi az első fóliát
```

eredménye lehet

```
((0 . "LAYER")                         Szimbólum típusa
  (2 . "0")                            Szimbólum neve
  (70 . 0)                             Flagek
  (62 . 7)                             Színszám, ha ki van kapcsolva, negatív
  (6 . "CONTINUOUS")                   Vonaltípus neve
)
```

Figyelje meg, hogy a fentiekben nem szerepel –1-es csoport. Az AutoCAD megjegyzi az egyes táblákból utoljára kiolvasott bejegyzést, és a **tblnext** függvény ugyanarra a táblára vonatkozó minden újabb hívásakor a következő bejegyzéssel tér vissza. Amikor egy tábla végigpásztázását elkezdi, egy nem `nil` értékű második argumentummal biztosítja a tábla visszapörgetését, hogy a függvény az első bejegyzéssel térjen vissza.

A blokk táblából kiolvasott bejegyzések egy –2-es csoportot, és a blokk definíciójában szereplő első rajzelem elemnevét tartalmazzák (amennyiben létezik ilyen). Azaz, ha adott egy DOBOZ nevű blokk,

```
(tblnext "block")                      Lekérdezi a blokk definícióját
```

eredménye lehet

```
((0 . "BLOCK")                         Szimbólum típusa
  (2 . "DOBOZ")                        Szimbólum neve
  (70 . 0)                             Flagek
  (10 9.0 2.0 0.0)                     Origó X,Y,Z koordinátái
  (-2 . <Elem neve: 40000126>)         Az első rajzelem
)
```

A –2-es csoportban szereplő elemnevet az **entget** és az **entnext** függvények elfogadják, de a többi rajzelemkezelő függvény nem. Például nem illesztheti be egy kiválasztási halmazba az **ssadd** függvény segítségével. A –2-es csoportban található elemnevet átadva az **entnext** függvénynek, végigpásztázhatja a blokkdefinícióban szereplő rajzelemeket; a blokkdefinícióban lévő utolsó elem után az **entnext** függvény `nil` értékkel tér vissza.

Megjegyzés Amennyiben a blokk nem tartalmaz rajzelemeket, a **tblnext** függvény eredményében szereplő –2-es csoport a blokk endblk elemének nevét tartalmazza.

tblobjname

Egy megadott szimbólumtábla-bejegyzés elemnevét adja vissza

(tblobjname *table-name symbol*)

A **tblobjname** függvény megkeresi a *table-name* nevű szimbólumtáblában a *symbol* nevű szimbólumot, és a szimbólumtábla bejegyzéséhez tartozó rajzelem nevével tér vissza.

A **tblobjname** függvény által visszadott rajzelem neveket az **entget** és **entmod** függvények is használhatják.

tblsearch

Egy szimbólumot keres egy szimbólumtáblában

(tblsearch *table-name symbol [setnext]*)

A **tblsearch** függvény megkeresi a *table-name* nevű szimbólumtáblában a *symbol* nevű szimbólumot. A függvény mindkét nevet automatikusan nagybetűvé konvertálja. Amikor a **tblsearch** függvény talált egy szimbólumot a megadott névvel, a függvény a **tblnext** függvénynél ismertetett formátumú listával tér vissza. Ha ilyet nem talál, `nil` értékkel tér vissza.

```
(tblsearch "style" "standard")  Lekérdez egy szövegstílust
```

eredménye lehet

```
((0 . "STYLE")                  Szimbólum neve
  (70 . 0)                      Flagek
  (40 . 0.0)                    Rögzített magasság
  (41 . 1.0)                    Szélességi tényező
  (50 . 0.0)                    Dőlésszög
  (71 . 0)                      Generálási flagek
  (3 . "txt")                   Elsődleges fontfájl
  (4 . "")                      Big-fontfájl
)
```

Normális esetben a **tblnext** függvény által lekérdezett táblabejegyzések sorrendjére a **tblsearch** függvény nincs kihatással. Ha azonban a **tblsearch** művelet sikeres volt, és a *setnext* argumentum jelen van `nil`-től különböző értékkel, akkor a **tblnext** bejegyzésszámlálója úgy módosul, hogy a következő **tblnext** hívás a jelen **tblsearch** hívással lekérdezett bejegyzés utáni bejegyzéssel fog visszatérni.

term_dialog

Valamennyi aktuális párbeszédpanelt lezárja, mintha a felhasználó mindegyiket elvetette volna

```
(term_dialog)
```

Amennyiben az AutoCAD egy alkalmazás befejeződésekor nyitott DCL fájlt talál, akkor automatikusan végrehajt egy **term_dialog** hívást. Ez a függvény főleg egymásba ágyazott párbeszédpanelek megszakítására használatos. A **term_dialog** függvény mindig `nil` értékkel tér vissza.

terpri

Egy újsor karaktert ír a parancssorba

```
(terpri)
```

A **terpri** függvényt nem lehet fájl I/O műveletre használni. Ha egy fájlba kíván újsor karaktert írni, akkor a **prin1**, **princ** vagy **print** függvényt használhatja.

textbox

Leméri egy adott szövegobjektum kiterjedését, és a szöveg köré rajzolható keret átlós koordinátáival tér vissza

```
(textbox elist)
```

Az *elist* argumentum egy, az **entget** függvény által visszaadott formátumú definíciós lista, amelynek egy szövegobjektumot kell definiálnia. Amennyiben az *elist* csak magát a szöveget definiáló mezőt tartalmazza, a többi szövegparaméter az aktuális (vagy alapértelmezés szerinti) beállításokat használja. Sikeres hívás esetén a **textbox** függvény két pontból álló listával tér vissza, máskülönben `nil` értékkel.

A **textbox** függvény által elfogadott minimális lista csak magát a szöveget tartalmazza.

`(textbox '((1 . "Helló világ.")))` *eredménye lehet:* ((0.0 0.0 0.0) (0.8 0.2 0.0))

Ebben az esetben a **textbox** függvény a hiányzó paraméterek értékeként az aktuális alapértelmezés szerinti értékeket használná fel.

A **textbox** függvény által visszaadott pontok a szövegobjektumot határoló képzeletbeli keretet úgy határozzák meg, mintha a szöveg beillesztési pontja a (0,0,0) pont, elforgatási szöge pedig 0 volna. Az első helyen visszaadott lista általában a (0.0 0.0 0.0) pont, hacsak a szövegobjektum nem dőlt, nem függőleges, vagy nem tartalmaz olyan

karaktereket, melyeknek alsó szára az alapvonal alá nyúlik (mint például a *g* vagy a *p* betű). Ez az első lista a szöveg köré rajzolható legkisebb téglalap bal alsó sarokpontjának a szöveg beillesztési pontjához viszonyított koordinátáit adja meg. A második lista ennek a téglalapnak a jobb felső sarokpontját jelöli. A függvény által visszaadott pontok mindig ennek a befoglaló keretnek a bal alsó és jobb felső sarokpontját határozzák meg, tekintet nélkül a vizsgált szöveg irányára.

textpage

Letörli az AutoCAD szöveges képernyőjét, és megjeleníti azt az egyképernyős AutoCAD konfigurációk esetében

```
(textpage)
```

A **textpage** függvény megegyezik a **textscr** függvénnyel, azzal a különbséggel, hogy ez a függvény a szöveges ablakból minden előzőleg megjelenített szöveget töröl. Visszatérési értéke mindig `nil`.

textscr

Egyképernyős rendszerek esetében a grafikus képernyőről átvált a szöveges képernyőre (hasonlóan, mint az AutoCAD Képváltó funkcióbillentyűje)

```
(textscr)
```

A **textscr** függvény visszatérési értéke mindig `nil`.

trace

Az AutoLISP hibakereső segédeszköze

```
(trace function...)
```

A **trace** függvény beállítja a *functions* argumentummal megadott függvények nyomkövető (trace) flagjét. A *function* argumentummal megadott függvény minden kiértékelésekor megjelenik a nyomkövető kijelzés, amely (a hívás mélységének szintjéig tabulálva) jelzi a függvény belépését, és kiírja a függvény eredményét.

```
(trace saját-függv)                eredménye  SAJÁT-FÜGGV
```

és beállítja a **SAJÁT-FÜGGV** nyomkövető flagjét. A **trace** függvény az utolsó függvény nevével tér vissza.

További tudnivalók

untrace függvény

trans

Egyik koordináta-rendszerből egy másikba transzformál egy pontot (vagy elmozdulást)

(trans *pt from to [disp]*)

A *pt* argumentum egy három valós értékből álló lista, amely jelenthet egy háromdimenziós pontot, vagy egy háromdimenziós elmozdulást (vektort). A *from* argumentum annak a koordináta-rendszernek a kódja, amelyben a *pt* meg van adva, a *to* pedig annak a a koordináta-rendszernek a kódja, amelybe a pontot transzformálni akarja. Az opcionális *disp* argumentum, amennyiben jelen van és értéke nem nil, azt jelzi, hogy a *pt* argumentumot nem pontként, hanem háromdimenziós elmozdulásként kell értelmezni. A *from* és a *to* argumentum lehet egész szám (az alábbi táblázatban megadott értékkel), rajzelemnév, vagy háromdimenziós kihúzási vektor.

Koordináta-rendszerek kódjai

Kód	*Koordináta-rendszer*
0	Világ (VKR)
1	Felhasználói (aktuális FKR)
2	Képernyő: A 0-ás vagy 1-es kóddal együtt használva az aktuális nézetablak KKR-e A 3-as kóddal együtt használva a modelltér aktuális nézetablakának KKR-e
3	A papírtér KKR-e (*csak* 2-es kóddal együtt használható)

Amennyiben a *from* és *to* argumentumként rajzelemnevet ad meg, akkor annak az **entnext**, **entlast**, **entsel**, **nentsel** és **ssname** függvények által visszaadott rajzelemnévvel megegyező formátumúnak kell lennie. Ennek segítségével egy adott rajzelem Objektum koordináta-rendszerébe (OKR), vagy megfordítva, abból egy másik koordináta-rendszerbe transzformálhat egy pontot. (Néhány rajzelem esetében az OKR megegyezik a VKR-rel, így ezeknél az OKR és VKR közötti konverzió egy üres műveletet jelent.) Egy háromdimenziós kihúzási irányvektor (három valós szám listája) egy másik módszer arra, hogy egy rajzelemet az OKR-ből, illetve az OKR-be

transzformáljon. Ez a módszer azon rajzelemek esetében azonban nem működik, amelyek OKR-je megegyezik a VKR-rel.

A **trans** függvény a *to* argumentum által megadott koordináta-rendszerben kifejezett háromdimenziós ponttal (vagy elmozdulással) tér vissza. Legyen adott például az az FKR, amely a VKR-nek Z tengely körüli 90 fokos, az óramutató járásával ellentétes irányú elforgatásával jön létre:

```
(trans '(1.0 2.0 3.0) 0 1) eredménye (2.0 -1.0 3.0)
(trans '(1.0 2.0 3.0) 1 0) eredménye (-2.0 1.0 3.0)
```

A koordináta-rendszerekről részletesebb ismertető található a "Koordináta-rendszerek közötti transzformációk" című résznél, a 251. oldalon .

Például, ha egy szöveg beillesztési pontjából kíván egy vonalat húzni (Traszter használata nélkül), akkor a szöveg beillesztési pontját annak saját OKR-éből át kell transzformálni az FKR-be.

```
(trans szöveg-beill-pont szöveg-elemnév 1)
```

Majd megadhatja az eredményt a Ponttól... promptnak.

Megfordítva, egy pontot (vagy elmozdulást) át kell transzformálnia a cél OKR-be, mielőtt az **entmod** függvénynek bemenő adatként átadná. Például, ha egy kört kíván elmozgatni (a MOZGAT parancs használata nélkül) az FKR-hez viszonyított (1,2,3) elmozdulásértékkel, akkor az elmozdulást először át kell transzformálnia az FKR-ből a kör OKR -ébe:

```
(trans '(1 2 3) 1 kör-elemnév)
```

Majd az eredményül kapott elmozdulás értéket kell hozzáadnia a kör középpontjának koordinátáihoz.

Például, ha van egy felhasználó által megadott pont, és meg akarja tudni, hogy az adott nézetben egy vonal melyik végpontja látszik ehhez a ponthoz közelebb, akkor a pontot át kell transzformálnia az FKR-ből a KKR-be.

```
(trans felhaszn-pont 1 2)
```

Majd a vonal mindkét végpontját a vonal OKR-éből a KKR-be kell transzformálnia.

```
(trans végpont vonal-elemnév 2)
```

Ennek alapján kiszámíthatók a felhasználó által megadott pont és a vonal végpontjai közötti távolságok (a *Z* összetevőket figyelmen kívül hagyva), és megállapítható, hogy melyik végpont látszik közelebbinek.

A **trans** függvény kétdimenziós pontok transzformációjára is felhasználható. Ezt a függvény a *Z* összetevő megfelelő értékkel való *feltöltésével* teszi meg. A használt *Z* összetevő értéke függ a `from`

argumentum által megadott kiinduló koordináta-rendszertől, valamint attól, hogy a transzformálni kívánt érték pont, avagy elmozdulás érték. Ha elmozdulás, akkor a *Z* érték mindig 0.0; ha pont, akkor a feltöltött *Z* meghatározása a következőképpen történik.

Z értékek kétdimenziós pontok transzformálásakor

From	*Feltöltött Z érték*
VKR	0.0
FKR	az aktuális kiemelés
EKR	0.0
KKR	Az aktuális szerkesztősíkra vetítve (az FKR *XY* síkja + az aktuális kiemelés)
PTKKR	Az aktuális szerkesztősíkra vetítve (az FKR *XY* síkja + az aktuális kiemelés)

type

A megadott tétel típusát adja vissza

`(type item)`

A visszaadott típus az alábbi atomok egyike lehet:

Szimbólumok típusai

Típus	*Jelentés*
REAL	Lebegőpontos számok
FILE	Fájlleírók
STR	Karakterláncok
INT	Egész számok
SYM	Szimbólumok
LIST	Listák (és felhasználói függvények)

Típus	*Jelentés*
SUBR	Belső függvények
EXSUBR	Külső (ADS) függvények
PICKSET	Kiválasztási halmazok
ENAME	Elemnevek
PAGETB	Függvények lapozási táblája

Azon tételek esetében, amelyek kiértékelése `nil` eredményt ad (például azon szimbólumok esetében, amelyekhez nincs érték rendelve) a függvény `nil` értékkel tér vissza.

Vegyük például az alábbi értékadó utasításokat:

```
(setq a 123 r 3.45 s "Helló!" x '(a b c))
(setq f (open "név" "r"))
```

ebben az esetben

```
(type 'a)          eredménye SYM
(type a)           eredménye INT
(type f)           eredménye FILE
(type r)           eredménye REAL
(type s)           eredménye STR
(type x)           eredménye LIST
(type +)           eredménye SUBR
(type nil)         eredménye nil
```

A következő példa a **type** függvényt használja:

```
(defun isint (a)
   (if (= (type a) 'INT)   egész TÍPUSÚ?
     T                     igen, eredmény T
     nil                   nem, eredmény nil
   )
)
```

unload_dialog

Töröl a memóriából egy DCL fájlt

(unload_dialog *dcl_id*)

Törli a memóriából a *dcl_id* (mely egy megelőző **new_dialog** hívásból származik) által azonosított DCL fájlt.

A visszatérési érték mindig `nil`.

untrace

Törli a megadott függvények nyomkövető flagjét

(untrace *function...*)

Az utolsó függvény nevével tér vissza.

A következő függvényhívás törli a **SAJÁT-FÜGGV** függvény nyomkövető flagjét:

```
(untrace saját-függv)          eredménye SAJÁT-FÜGGV
```

További tudnivalók

trace függvény

vector_image

Egy vektort rajzol az éppen aktív párbeszédpanel-ábrába

(vector_image *x1 y1 x2 y2 color*)

Ez a függvény egy, az (*x1,y1*) pontból az (*x2,y2*) pontba tartó vektort rajzol az éppen aktív (a **start_image** által megnyitott) párbeszédpanel-ábrába. A *color* paraméter egy AutoCAD színszám, vagy az alábbi logikai színszámok egyike:

A color attribútum szimbolikus nevei

Színszám	*ADI mnemonik*	*Jelentés*
–2	BGLCOLOR	A háttér aktuális színe az AutoCAD grafikus képernyőjén
–15	DBGLCOLOR	A háttér aktuális színe a párbeszédpanelben
–16	DFGLCOLOR	Az előtér aktuális színe a párbeszédpanelben (szövegek számára)
–18	LINELCOLOR	A vonalak aktuális színe a párbeszédpanelben

Az origó (0,0) az ábra bal felső sarka. A jobb alsó sarok koordinátáit a méretfüggvények (**dimx_tile** és **dimy_tile**) hívásával kaphatja meg.

ver

Az AutoLISP aktuális verziószámát tartalmazó karakterlánccal tér vissza

(ver)

A **ver** függvényt (az **equal** függvénnyel) a programok kompatibilitásának ellenőrzésére használhatja. A karakterlánc formátuma a következő:

```
"AutoLISP Release X.X (nn)"
```

ahol az *X.X* az éppen használt verzió száma, és *nn* egy két betűből álló nyelvleíró.

(ver) *lehetséges eredménye* `"AutoLISP Release 13.0 (en)"`

Az alábbiakban a kétbetűs nyelvleíróra láthatunk néhány példát:

(en) am./ang. (es) spanyol (fr) francia
(de) német (it) olasz

vmon

Virtuális függvénylapozást tesz lehetővé (elavult)

```
(vmon)
```

Erre a függvényre már nincs szükség, azonban továbbra is érvényes a korábbi verziókkal való kompatibilitás megőrzése céljából.

További tudnivalók

"Virtuális függvénylapozás" című résznél, a 567. oldalon

vports

Az aktuális nézetablak-konfiguráció nézetablak-leíróinak listájával tér vissza

```
(vports)
```

Mindegyik nézetablak-leíró a nézetablak azonosítószámát, valamint a nézetablak bal alsó és jobb felső sarokpontjának pozícióját tartalmazza.

Amennyiben az AutoCAD TILEMODE rendszerváltozója 1-re (be) van állítva, a lista az AutoCAD NÉZETABL parancsával létrehozott nézetablak-konfigurációt írja le. A nézetablak sarkait 0.0 és 1.0 közötti értékekkel fejezi ki, ahol a (0.0, 0.0) a képernyő grafikus területének bal alsó, az (1.0, 1.0) pedig a jobb felső sarkát képviseli. Ha a TILEMODE értéke 0 (ki), akkor a visszatérő lista az MNÉZET paranccsal létrehozott nézetablak rajzelemek leírását tartalmazza. A nézetablak rajzelemek sarokpontjai papírtérbeli koordinátákban vannak kifejezve. Kikapcsolt (0) TILEMODE esetén az 1-es számú nézetablak mindig a papírtérben van.

Például egyetlen nézetablakból álló konfiguráció és a TILEMODE 1 értéke esetén a **vports** függvény az alábbi eredménnyel térhet vissza:

```
((1 (0.0 0.0) (1.0 1.0)))
```

Hasonlóképpen, ha adott a képernyő négy sarkában elhelyezkedő négy egyenlő nagyságú nézetablakból álló konfiguráció, és a

TILEMODE be van kapcsolva, akkor a **vports** függvény a következő eredménnyel térhet vissza:

```
( (5 (0.5 0.0) (1.0 0.5))
  (2 (0.5 0.5) (1.0 1.0))
  (3 (0.0 0.5) (0.5 1.0))
  (4 (0.0 0.0) (0.5 0.5)) )
```

A listában mindig az aktuális nézetablak leírója szerepel az első helyen. A fenti példában az 5-ös számú az aktuális nézetablak.

wcmatch

Globális karakter-összehasonlítást végez egy karakterláncon

(wcmatch *string pattern*)

A **wcmatch** függvény egyezést keresve összehasonlítja a *string* karakterláncot és a *pattern* (minta) argumentumot. Egyezés esetén a visszatérési érték T, egyébként nil. A *string* és a *pattern* egyaránt lehet idézőjeles karakterlánc vagy változó. A *pattern* az alábbi globális összehasonlító karaktereket tartalmazhatja. A *string* és a *pattern* argumentumoknak csak az első, megközelítőleg 500 karaktere kerül összehasonlításra, a továbbiakról a függvény nem vesz tudomást.

Globális karakterek

Karakter	*Meghatározás*
# (font)	Egyetlen számjegyet helyettesít
@ (at)	Egyetlen alfabetikus karaktert helyettesít
. (pont)	Egyetlen nem-alfanumerikus karaktert helyettesít
* (csillag)	Bármilyen karaktersorozatot helyettesít, az üreset is. A keresési minta elején, közepén és végén is használható
? (kérdőjel)	Egyetlen karaktert helyettesít
~ (hullám)	Ha ez a minta első karaktere, akkor a mintával egyezők *kivételével* az összeset keressük
[...]	A szögletes zárójelek között felsorolt karakterek bármelyikét (*egyet*) helyettesíti
[~...]	A szögletes zárójelek között fel *nem* sorolt karakterek bármelyikét helyettesíti

Globális karakterek (*folytatás*)

Karakter	*Meghatározás*
- (kötőjel)	Szögletes zárójelek között használatos, segítségével megadható egy karakter lehetséges értékeinek halmaza
, (vessző)	Elválaszt két mintát
` (ford. aposztróf)	Érvényteleníti a speciális karakterekhez rendelt funkciót (betű szerint értelmezi a következő karaktert)

`(wcmatch "Neve" "N*")` *eredménye* `T`

Ezzel ellenőrizheti, hogy a `Neve` karakterlánc `N` betűvel kezdődik-e. A mintában vesszőket is használhat, ezzel egyszerre több feltételt adva meg. Az alábbi példa három összehasonlítást végez:

`(wcmatch "Neve" "???,~*v*,N*")` *eredménye* `T`

Abban az esetben, ha a három feltétel bármelyike teljesül, a **`wcmatch`** függvény `T` értékkel tér vissza. A fenti példában a feltételek a következők : `Neve` három karakterből áll (hamis); `Neve` nem tartalmaz `v` betűt (hamis), és `Neve` első betűje `N` (igaz). Legalább egy feltétel teljesült, ezért a visszatérési érték `T`.

Az összehasonlítás megkülönbözteti a kis- és nagybetűket is, így a minta megadásakor erre is tekintettel kell lennie. A *`string`* és a *`pattern`* argumentumban AutoLISP függvények visszatérési értékeinek és változóknak a használata is megengedett.

Ha egy globális karakter előfordulását szeretné megvizsgálni egy karakterláncban, akkor egy fordított aposztróf (`) jellel *kiemelve* teheti meg. *A kiemelés* azt jelenti, hogy a fordított aposztrófot követő globális karaktert betű szerint kell értelmezni, figyelmen kívül hagyva annak speciális jelentését. Például, ha egy vesszőt keres bárhol a `Neve` karakterláncban, billentyűzze be a következőt:

`(wcmatch "Neve" "*`,*")` *eredménye* `nil`

Megjegyzés Mivel a következő AutoLISP verziókban a globális karakterek készlete kibővülhet, a kompatibilitás megőrzése érdekében célszerű az összes nem-alfanumerikus karaktert kiemelni a mintában.

A visszahajló törtjelet (\) a C programnyelv és az AutoLISP is vezérlő (escape) karakterként használja, ezért két (\\) jelet kell tegyen, ha a karakterláncban egy visszahajló törtjelet kíván szerepeltetni. Például,

ha egy visszahajló törtjelet keres bárhol a `Neve` karakterláncban, ezt billentyűzze be:

`(wcmatch "Neve" "*`*")` *eredménye* `nil`

Minden szögletes zárójelbe tett karakter ([. . .]) betű szerint értendő, így ezek kiemelése nem szükséges, az alábbiak kivételével: a hullám jel (~) csak akkor értendő betű szerint, ha *nem* az első karakter a zárójelen belül (például: `"[A~BC]"`), egyébként negáló karakterként működik, és bármely, a zárójelek között *nem* található karakterrel való egyezést keres (pédául: `"[~ABC]"`). A kötőjelnek (–) a zárójelen belül akkor van betű szerinti értelmezése, ha ez az első vagy utolsó karakter a zárójelen belül (például: `"[-ABC]"` vagy`"[ABC-]"`), vagy közvetlenül a bevezető hullám utám következik (például: `"[~-ABC]"`). Ellenkező esetben a szögletes zárójelen belül a kötőjel (–) annak meghatározására használható, hogy egy adott karakternek milyen értéktartományon belül kell elhelyezkednie. A tartomány meghatározása csak különálló karakterek esetében lehetséges, így a `"SZO[1-38]"` a `SZO1`, `SZO2`, `SZO3` és `SZO8` karakterláncokra illik rá, az `"[A-Z]"` mintát pedig az összes különálló nagybetű kielégíti.

A lezáró szögletes zárójel (`"]"`) szintén betű szerint értendő a zárójeleken belül, ha ez a zárójelben lévő első karakter, vagy ha közvetlenül a bevezető hullám jelet követi (például: `"[ ]ABC]"` vagy `"[~]ABC]"`).

while

Kiértékeli a feltétel kifejezést, és ha annak értéke nem `nil`, akkor a többi kifejezést is kiértékeli; addig ismétli ezt a folyamatot, amíg a feltétel kifejezés értéke `nil` nem lesz

`(while` *`testexpr expr...`*`)`

A **`while`** függvény addig folytatja a kiértékelést, amíg a *`testexpr`* kifejezés értéke `nil` nem lesz; majd az utolsó *`expr`* kifejezés legutóbbi értékével tér vissza.

A következő programrészlet tíz alkalommal hívja meg a felhasználó által definiált **`VALAMI-FÜGGV`** függvényt, a `test` 1-től 10-ig terjedő értékeivel. Ezt követően az utolsó kiértékelt kifejezés értékével, 11-gyel tér vissza.

```
(setq test 1)
(while (<= test 10)
  (some-func test)
  (setq test (1+ test))
)
```

write-char

Egy karaktert ír a képernyőre vagy egy nyitott fájlba

(write-char *num [file-desc]*)

A *num* argumentum a kiírni kívánt karakter decimális ASCII kódja; a függvény ezzel a kóddal tér vissza.

`(write-char 67)` *eredménye* `67`

és megjeleníti a képernyőn a C betűt. Feltételezve, hogy az `f` egy megnyitott fájl azonosítója:

`(write-char 67 f)` *eredménye* `67`

és a `C`-t ebbe a fájlba írja ki.

Azok az operációs rendszerek, amelyek alatt az AutoCAD futtatható, eltérő módon jelölik az ASCII fájlok sorvégeit. A UNIX rendszerek például egyetlen újsor karaktert használnak (LF, ASCII kódja 10), míg a DOS rendszerek ugyanerre a célra egy karakterpárt (CR/LF, ASCII kódjuk 13 és 10) használnak. Az AutoLISP programok fejlesztésének megkönnyítése érdekében, a **write-char** függvény az újsor karaktert (ASCII kódja 10) az éppen használt operációs rendszernek megfelelő "sor vége" karakterré (vagy karaktersorozattá) alakítja át. Tehát a DOS operációs rendszer esetében

`(write-char 10 f)` *eredménye* `10`

de a fájlba a CR/LF (ASCII kódjuk 13 és 10) karakterpárt írja. A **write-char** függvény NUL karaktert (ASCII kódja 0) nem képes a fájlba írni .

További tudnivalók

"ASCII kódok" című A melléklet

write-line

Egy karakterláncot ír a képernyőre vagy egy nyitott fájlba

(write-line *string [file-desc]*)

A függvény a szokott módon idézőjelek közé tett *string* karakterlánccal tér vissza, de a fájlba az idézőjeleket nem illeszti be. Példaként tételezzük fel, hogy az `f` egy nyitott fájl érvényes azonosítója:

`(write-line "Próba" f)` *kiírás* `Próba` *és eredménye* `"Próba"`

xdroom

A rajzelem számára elérhető, bővített adatokat (Xdata) tartalmazó hely méretével tér vissza

(xdroom *ename*)

Sikeres hívás esetén az **xdroom** függvény visszatérési értéke `nil`. Mivel egy rajzelemhez korlátozott (jelenleg 16 kilobájt) mennyiségű bővített adat tartozhat, továbbá több alkalmazás is fűzhet bővített adatokat ugyanahhoz a rajzelemhez, ezért ez a függvény arra szolgál, hogy segítségével az alkalmazások ellenőrizhessék, van-e elegendő hely az adott rajzelemhez általuk hozzáfűzni kívánt bővített adatok számára. Az **xdroom** hívása kapcsolódhat az **xdsize** függvényéhez, amely egy bővített adatlista méretét adja vissza.

Az alábbi példa egy nézetablak-rajzelem bővített adatai számára rendelkezésre álló helyet vizsgálja meg. Tételezzük fel, hogy a `nabneve` változó egy nézetablak-rajzelem nevét tartalmazza:

```
(xdroom nabneve)          eredménye  16162
```

Ebben a példában a bővített rajzelemadatok eredeti 16383 bájtjából 16162 áll rendelkezésre, ami azt jelenti, hogy 221 bájt már foglalt. Az **xdsize** függvény segítségével meghatározhatja a rendelkezésre álló bővített adathelyet.

xdsize

Annak a helynek a méretével tér vissza (bájtokban), amelyet egy lista elfoglalna, ha egy rajzelemhez bővített adatként hozzá lenne csatolva

(xdsize *list*)

Sikertelen hívás esetén a függvény `nil` értékkel tér vissza. A *list* argumentumnak érvényes bővített adatlistának kell lennie, tartalmaznia kell egy --előzőleg a **regapp** függvénnyel bejegyzett-- alkalmazásnevet és egyenlő számú nyitó és záró határoló mezőt (csoportkódja 1002). Érvénytelen *list* argumentum hibát okoz, és a hibának megfelelő kód az ERRNO rendszerváltozóba kerül. Amennyiben a bővített adatok be nem jegyzett alkalmazásnevet tartalmaznak, a következő hibaüzenet jelenik meg (feltéve, hogy a CMDECHO be van kapcsolva):

Érvénytelen alkalmazásnév a(z) 1001 csoportban

A *list* argumentum kezdődhet –3-as csoportkóddal (bővített adat jelzője), de ez nem feltétlenül szükséges. Mivel a bővített adatok több

alkalmazásról is tartalmazhatnak információt, a listát még egyszer zárójelbe kell tenni.

```
(-3 ("SAJALK" (1000 . "SODRONYING")
              (1002 . "{")
              (1040 . 0.0)
              (1040 . 1.0)
              (1002 . "}")
    )
)
```

Íme ugyanez a példa a –3-as csoportkód nélkül. Ez a lista az első listának csak a **cdr** értéke, de fontos, hogy a zárójeleket ez is tartalmazza:

```
( ("SAJALK" (1000 . "SODRONYING")
            (1002 . "{")
            (1040 . 0.0)
            (1040 . 1.0)
            (1002 . "}")
    )
)
```

xload

Betölt egy ADS alkalmazást

(xload *application [onfailure]*)

Az *application* argumentum megadható a futtatható fájl nevét tartalmazó, idézőjelbe tett karakterlánccal, vagy változóval. A fájl betöltésekor az AutoLISP ellenőrzi, hogy az ADS alkalmazás érvényes-e, valamint kompatibilis-e az aktuális ADS rendszerrel és az éppen használt AutoLISP verzióval.

Amennyiben az **xload** művelet sikertelen, általában egy AutoLISP hibát okoz. Ha azonban az *onfailure* argumentumot is megadta, akkor az **xload** függvény a hibajelzés helyett ennek az argumentumnak az értékével tér vissza.

Amennyiben az alkalmazás betöltése sikeres volt, a függvény annak nevével tér vissza.

`(xload "/sajat/alk")` *ha sikeres, eredménye* `"/sajat/alk"`

Amennyiben egy olyan programot akar betölteni, amelyik már a memóriában van, akkor az **xload** függvény a következő üzenetet jeleníti meg:

"*alkalmazás*" alkalmazás már be van töltve.

és az alkalmazás nevével tér vissza. Az éppen betöltött ADS alkalmazásokat az **xload** hívása előtt az **ads** függvénnyel ellenőrizheti.

xunload

Töröl egy ADS alkalmazást a memóriából

(xunload *application [onfailure]*)

Amennyiben az alkalmazás törlése sikeres volt, annak nevével tér vissza, egyébként hibaüzenetet küld.

Adja meg az *application* argumentumot mint idézőjelbe tett karakterláncot, vagy mint változót, amely az **xload** függvénnyel betöltött alkalmazás nevét tartalmazza. Az alkalmazás nevének pontosan meg kell egyeznie azzal, amit az alkalmazás betöltésekor megadott. Az **xload** függvényhívásban használt elérési útvonal (könyvtárnév) elhagyható.

A következő függvényhívás például törli az **xload** függvénnyel korábban betöltött alkalmazást:

`(xunload "alk")` *ha sikeres, eredménye* `"alk"`

Amennyiben az **xunload** művelet sikertelen, általában egy AutoLISP hibát okoz. Ha azonban az *onfailure* argumentumot is megadta, akkor az **xunload** függvény a hibajelzés helyett ennek az argumentumnak az értékével tér vissza. Az **xunload** függvénynek ez a tulajdonsága megegyezik az **xload** függvényével.

zerop

Ellenőrzi, hogy egy szám értéke nulla-e

(zerop *number*)

Ha a *number* argumentum értéke nulla a visszatérési érték `T`; egyébként pedig `nil`.

```
(zerop 0)          eredménye T
(zerop 0.0)        eredménye T
(zerop 0.0001)     eredménye nil
```

14

A KÜLSŐLEG DEFINIÁLT PARANCSOK ÉS RENDSZERVÁLTOZÓK HASZNÁLATA

A *külsőleg definiált* jelző azokra az AutoCAD parancsokra és rendszerváltozókra vonatkozik, amelyeket az ADS, ARX vagy AutoLISP alkalmazások definiálnak. Az AutoLISP alkalmazások a külsőleg definiált parancsokat a beépített AutoCAD parancsoktól eltérő módon érik el. Az AutoLISP alkalmazások a `command` függvény segítségével érik el a beépített AutoCAD parancsokat, míg a külsőleg definiált parancsok vagy parancscsoportok saját programozási felülettel rendelkeznek, amely lehetővé teszi az AutoLISP alkalmazások számára, hogy kihasználják azok funkcióit.

Az alább felsorolt parancsokat ismertetjük ebben a fejezetben. További információt az AutoCAD on-line súgójában találhat.

3DSBE	FÉNY	PSBE	RIEDGE
3DSKI	FORGAT3D	PSKÉP	RIGAMUT
ANYAGTÁR	GIFBE	PSKITÖLT	RIGREY
ASEADMIN	ILLESZT	RANY	RITHRESH
ASEEXPORT	JELENET	RBEÁLL	RKONFIG
ASELINKS	KAL	RENDER	RSTAT
ASEROWS	KÉPMENTŐ	RENDERKÉP	TIFFBE
ASESELECT	KÉPÚJRA	RIASPECT	TÜKRÖZ3D
ASESQLED	PCXBE	RIBACKG	VISLKONV

3DSBE

Egy 3D Studio fájlt (*.3ds*) importál

(c:3dsbe *mode [multimat create] file*)

A `mode` argumentum egy egész szám, amely megadja, hogy a parancsot interaktív (`mode` = 1) vagy nem interaktív (`mode` = 0) módon kezelje a rendszer. Ha a `mode` értéke 0, a `file` argumentum mellett a `multimat` és a `create` argumentumot is meg kell adni. Amennyiben a `mode` értéke 1, csak a `file` argumentumot kell megadni. A `file` argumentum egy karakterlánc, amely az importálni kívánt *.3ds* fájlt adja meg. Szükséges a *.3ds* kiterjesztés.

A következő függvényhívás importálás céjából megnyitja a *globe.3ds* 3D Studio fájlt, majd kéri a felhasználótól az importálás módját meghatározó adatokat:

```
(c:3dsbe 1 "globe.3ds")
```

A `multimat` és a `create` argumentumok egész számok, és megadják, hogy a függvény hogyan kezelje a több anyaggal rendelkező objektumokat, illetve hogyan hozzon létre új objektumokat.

Ha a függvényt a mode 0 (nem interaktív) értékével használja, akkor az argumentumok megadják, hogyan kezelje a függvény a több anyagot, és hogyan rendezze el az új objektumokat. Ez a mód mindig az összes objektumot importálja a *.3ds* fájlba.

A multimat és a create argumentum értékei

Argumentum	*Leírás*
multimat	Megadja, hogyan kezelje a függvény a több anyaggal rendelkező objektumokat: 0 —Mindegyik anyag számára új fóliát hoz létre 1 — Az első anyagot az új objektumhoz rendeli
create	Megadja, hogy a függvény hogyan hozzon létre új objektumot: 0—Minden 3DS objektum számára létrehoz egy fóliát 1—Minden 3DS szín számára létrehoz egy fóliát 2—Minden 3DS anyag számára létrehoz egy fóliát 3—Az összes objektumot egyetlen fóliára helyezi

A következő függvényhívás felhasználói adatbevitel nélkül importálja a *globe.3ds* fájl összes objektumát, a több anyaggal rendelkező objektumokat felosztja, és minden új objektumot ugyanarra a fóliára helyez:

```
(c:3dsbe 0 0 3 "globe.3ds")
```

Külsőleg definiált függvény

render ARX alkalmazás

3DSKI

Egy 3D Studio fájlt exportál

(c:3dski *sset omode div smooth weld file*)

A **c:3dsout** függvénynek hat argumentuma van. Az *sset* argumentum egy kiválasztási halmaz, amely az exportálni kívánt AutoCAD objektumokat tartalmazza. Az *omode* argumentum egy egész szám, amely az AutoCAD adatok ábrázolási módját adja meg. Ha az *omode* értéke 0, a kimenet DXF, ha pedig értéke 1, akkor Visual Link formátumú. A *div* argumentum egy egész szám, amely megadja, hogy miképp történjen az AutoCAD objektum 3D Studio objektummá való felosztása. Ha a *div* értéke 0, minden egyes AutoCAD fólia számára létrehoz egy objektumot. Amennyiben a *div* értéke 1, minden AutoCAD szín számára létrehoz egy objektumot, ha pedig 2, minden AutoCAD objektumtípus számára létrehoz egy objektumot. A *smooth* argumentum egy egész szám, amely megadja az automatikus simítás küszöbszögét. Amennyiben a *smooth* értéke –1, akkor nincs automatikus simítás; ha értéke 0 és 360 között van, akkor az AutoCAD simítást hajt végre abban az esetben, ha a felületi normálisok közötti szög kisebb ennél az értéknél. A *weld* argumentum egy valós szám, amely megadja a szomszédos csomópontok összeolvasztási küszöbtávolságát. Ha *weld* értéke negatív, akkor az összeolvasztás ki van kapcsolva, ha értéke 0 vagy annál nagyobb érték, akkor az AutoCAD összeolvasztja az ennél az értéknél közelebb fekvő csomópontokat. Az utolsó argumentum, *file*, a létrehozandó 3D Studio fájl neve. A *.3ds* kiterjesztést meg kell adni.

Az alábbi függvényhívás az egész rajzot exportálja, létrehozza a rajz fóliájára alapuló 3D Studio objektumokat úgy, hogy 30 fokos simítási szöget és 0.1 összeolvasztási távolságot használ:

```
(c:3dsout (ssget "X") 0 0 30 0.1 "testav.3ds")
```

Külsőleg definiált függvény

render ARX alkalmazás

ANYAGTÁR

Anyagkönyvtárakat kezel

(c:anyagtár *mode name [file]*)

Az alábbi táblázatban bemutatott `mode` argumentum egy karakterlánc, amely magadja a függvény által végrehajtott műveletet. A `name` argumentum egy karakterlánc, amely megadja az exportálni, importálni illetve törölni kívánt anyag nevét. Az opcionális `file` argumentum pedig olyan karakterlánc, amely megadja az anyagkönyvtár fájl nevét. A `file` argumentumnak tartalmaznia kell a *.mli* kiterjesztést.

Az ANYAGTÁR függvény módjai

Mode	*Leírás*
I	Anyagot importál egy könyvtárból
E	Anyagot exportál egy könyvtárba
D	Anyagot töröl egy rajzból

Például:

```
(c:anyagtár "I" "brass" "render.mli")
```

a BRASS nevű anyagot importálja az AutoCAD *render.mli* nevű Render anyagkönyvtárából.

A `file` argumentumot nem lehet a Delete módban használni:

```
(c:anyagtár "D" "fém")
```

Külsőleg definiált függvény

render ARX alkalmazás

ASEADMIN

Adminisztratív függvényeket hajt végre külső adatbázist kezelő parancsok számára

(c:aseadmin *arg1 arg2 ...*)

Az argumentumok sorrendje, száma és típusa megegyezik azzal, amit a Parancs prompt után billentyűzne be.

Külsőleg definiált függvény

ase ARXalkalmazás

ASEEXPORT

Kiválasztott objektumok csatolási információit exportálja

```
(c:aseexport arg1 arg2 ...)
```

Az argumentumok sorrendje, száma és típusa megegyezik azzal, amit a Parancs prompt után billentyűzne be.

Külsőleg definiált függvény

ase ARX alkalmazás

ASELINKS

Objektumok és külső adatbázisok közötti csatolások kezelését végzi

```
(c:aselinks arg1 arg2 ...)
```

Az argumentumok sorrendje, száma és típusa megegyezik azzal, amit a Parancs prompt után billentyűzne be.

Külsőleg definiált függvény

ase ARX alkalmazás

ASEROWS

Külső adatbázis-táblák adatait kezeli

```
(c:aserows arg1 arg2 ...)
```

Az argumentumok sorrendje, száma és típusa megegyezik azzal, amit a Parancs prompt után billentyűzne be.

Külsőleg definiált függvény

ase ARX alkalmazás

ASESELECT

Szöveges és gafikus kiválasztási halmazokból hoz létre egy új kiválasztási halmazt

```
(c:aseselect arg1 arg2 ...)
```

Az argumentumok sorrendje, száma és típusa megegyezik azzal, amit a Parancs prompt után billentyűzne be.

Külsőleg definiált függvény

ase ARX alkalmazás

ASESQLED

SQL utasításokat hajt végre

```
(c:asesqled arg1 arg2 ...)
```

Az argumentumok sorrendje, száma és típusa megegyezik azzal, amit a Parancs prompt után billentyűzne be.

Külsőleg definiált függvény

ase ARX alkalmazás

FÉNY

Fényforrásokat, illetve fényeffektusokat hoz létre; módosítja vagy törli azokat

```
(c:fény mode [options])
```

A FÉNY parancs segítségével új fényforrásokat hozhat létre, illetve létező fényforrásokat módosíthat vagy törölhet. A parancsot megvalósító függvénynek az alábbi táblában bemutatott nyolc módja létezik. Mindegyik módot egy karakterlánc-argumentum határozza meg; az `option` argumentumok a `mode` értékétől függnek.

A FÉNY függvény módjai

Mód	*Leírás*
A	Háttérvilágítás intenzitásának beállítása és lekérdezése
D	Létező fényforrás törlése
L	A rajzban lévő fényforrások listája, illetve a megnevezett fényforrás adatai

A FÉNY függvény módjai (*(folytatás)*)

Mód	*Leírás*
M	Létező fényforrás módosítása
ND	Új távoli fényforrás létrehozása
NP	Új pontszerű fényforrás létrehozása
NS	Új reflektor létrehozása
R	Létező fényforrás átnevezése

Megjegyzés Ez a parancs a papírtérben *nem* megengedett.

A—Ambient (Szórt háttérvilágítás)

Beállítja vagy lekérdezi a szórt háttérvilágítás intenzitását.

(c:fény "A" *[intensity [color]]*)

Az *intensity* argumentum egy 0.0 és 1.0 közötti valós szám; ha az *intensity* argumentum elmarad, értéke alapértelmezés szerint 1.0 lesz. A *color* argumentum egy lista, amely egy RGB színhármast ad meg; ha elmarad, értéke alapértelmezés szerint (1.0 1.0 1.0).

A következő sor például a szórt háttérvilágítás intenzitását 0.6-ra állítja,

```
(c:fény "A" 0.6)
```

Az aktuális szórt háttérvilágítás intenzitásának lekérdezése esetén ne adja meg az *intensity* argumentumot.

`(c:fény "A")` *eredménye* `0.3` *(vagy más valós érték)*

D—Delete (Töröl)

Létező fényforrást töröl.

(c:fény "D" *name*)

A *name* argumentum egy karakterlánc, amely a törölni kívánt fényforrás nevét adja meg.

```
(c:fény "D" "RÉGIFÉNY")
```

L—List (Listáz)

Kilistázza a rajzban lévő fényforrásokat, vagy visszaadja a megnevezett fényforrás adatait.

(c:fény "L" *[name]*)

A *name* argumentum egy karakterlánc, amely a kilistázni kívánt fényforrás nevét adja meg. Ha a *name* argumentum elmarad, a **c:fény** függvény a rajzban definiált összes fényforrást kilistázza. Amennyiben megadja a *name* argumentumot, a **c:fény** függvény visszaadja a megnevezett fényforrás adatait.

```
(c:fény "L")                    eredménye ("REFL1")
(c:fény "L" "REFL1")            eredménye:
("S" <Elem neve: 600039c6> 157.22 (-97.8002 1.21954e-14
70.0222) (13.8456 20.103 0.0) (0.944 0.89824 0.736) 0 12.0
16.0 3.0 "ON")
```

M—Modify (Módosít)

Létező fényforrást módosít.

(c:fény "M" *name [intensity [from [to [color*
[nil *[hotspot [falloff* [nil [nil [nil [nil*
[nil [nil [nil [nil [nil [nil*
***[attenuation]]]]]]]]]]]]]]])**

A Modify mód argumentumai a következő táblázatban szerepelnek:

FÉNY "M" mód argumentumai

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
name	STR	Egyedi fényforrásnév	Nincs
intensity	REAL	Valós szám 0-tól az alapértelmezés szerinti maximumig	A tompuláson alapszik
from	LIST	A fényforrás helye	Az aktuális nézőpont
to	LIST	A fényforrás célpontja	Az aktuális nézet célpontja
color	LIST	Bármilyen RGB színhármas	1.0, 1.0, 1.0
hotspot	REAL	A fényesség sugarának szöge fokokban (1 és 160 közöti érték)	44.0
falloff	REAL	A gyors gyengülés területet fokokban kifejező szög (1 és 160 közötti érték, a hotspot értéknél nagyobbnak kell lennie)	45.0

FÉNY "M" mód argumentumai (*folytatás*)

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
attenuation	INT	0 = nincs tompulás 1 = inverz lineáris tompulás 2 = inverz négyzetes tompulás	1

Ha egy argumentum értéke `nil`, vagy elmarad az argumentumlista végéről, akkor megőrzi aktuális értékét. Amennyiben egy argumentum nem vonatkozik módosítandó fényforrás-típusra, akkor adjon meg `nil` értéket helyette — vagy hagyja el, ha az az argumentumlista végén szerepel.

A következő függvényhívás például kékre változtatja a D1 nevű távoli fényforrás színét .

```
(c:fény "M" "D1" nil nil nil '(0.0 0.0 1.0))
```

ND—New Distant Light (Új távoli fényforrás)

Egy új távoli fényforrást hoz létre.

```
(c:fény "ND" name [intensity [from [to [color ]]]])
```

A New Distant Light mód argumentumait a következő táblázat tartalmazza:

A FÉNY "ND" mód argumentumai

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
name	STR	Egyedi fényforrásnév	Nincs
intensity	REAL	Valós szám 0-tól az alapértelmezés szerinti maximumig	A tompuláson alapszik
from	LIST	A fényforrás helye	Az aktuális nézőpont
to	LIST	A fényforrás célpontja	Az aktuális nézet célpontja
color	LIST	Bármilyen RGB színhármas	1.0, 1.0, 1.0

NP—New Point Light (Új pontszerű fényforrás)

Egy új pontszerű fényforrást hoz létre.

(c:fény "NP" *name [intensity [from [nil [color [nil [nil [nil [nil [attenuation]]]]]]]]]*)

A New Point Light mód argumentumait a következő táblázat tartalmazza:.

A FÉNY—"NP" mód argumentumai

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
name	STR	Egyedi fényforrásnév	Nincs
intensity	REAL	Valós szám 0-tól az alapértelmezés szerinti maximumig	A tompuláson alapszik
from	LIST	A fényforrás helye	Az aktuális nézőpont
color	LIST	Bármilyen RGB színhármas	1.0, 1.0, 1.0
attenuation	INT	0 = nincs tompulás 1 = inverz lineáris tompulás 2 = inverz négyzetes tompulás	1

Három argumentum— a `to`, a `hotspot` és a `falloff`— nem befolyásolja a pontszerű fényforrást. Ezeket `nil` értékkel kell átadnia, mikor egy új pontszerű fényforrást hoz létre.

Az alábbi függvényhívás például egy ÚJPT1 nevű pontszerű fényforrást hoz létre .

```
(c:fény "NP" "ÚJPT1")
```

Az ÚJPT1 fényforrás az alapértelmezés szerinti fényerősséggel, az aktuális tompulási beállítással, az alapértelmezés szerinti nézőpont-hellyel és az alapértelmezés szerinti fehér színnel rendelkezik.

Megjegyzés A pontszerű fényforrások esetében az alapértelmezés szerinti maximális intenzitás függ az aktuális pontszerű/reflektor tompulás beállításától. Ha nincs tompulás, ez az érték 1.00; inverz lineáris tompulás esetén az érték a rajz maximális kiterjedése; inverz négyzetes tompulás esetén pedig a maximális kiterjedés négyzete.

NS—New Spotlight (Új reflektor)

Egy új reflektort hoz létre.

```
(c:fény "NS" name [intensity [from [to [color [nil
[hotspot [falloff [nil [nil [attenuation]]]]]]]]])
```

A New Spotlight mód argumentumai a következő táblázatban találhatók:

A FÉNY "NS" mód argumentumai

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
name	STR	Egyedi fényforrásnév	Nincs
intensity	REAL	Valós szám 0-tól az alapértelmezés szerinti maximumig	A tompuláson alapszik
from	LIST	A fényforrás helye	Az aktuális nézőpont
to	LIST	A fényforrás célpontja	Az aktuális nézet célpontja
color	LIST	Bármilyen RGB színhármas	1.0, 1.0, 1.0
hotspot	REAL	A fényesség sugarának szöge fokokban (1 és 160 közöti érték)	44.0
falloff	REAL	A gyors gyengülés területet fokokban kifejező szög (1 és 160 közötti érték, a hotspot értéknél nagyobbnak kell lennie)	45.0
attenuation	INT	0 = nincs tompulás 1 = inverz lineáris tompulás 2 = inverz négyzetes tompulás	1

A következő példa egy ÚJREFL nevű reflektort hoz létre.

```
(c:fény "NS" "ÚJREFL" 137.82 '(12.0 6.0 24.0)
  '(78.0 78.0 24.0) nil 30.0 32.0)
```

Az ÚJREFL egy 137.82 intenzitású reflektor. Színe az alapértelmezés szerinti (fehér). A reflektor helye (12,6,24), célpontjának helye pedig (78,78,24). Kúpszöge 32 fok széles, a hotspot 30 fokos.

Megjegyzés A reflektorok esetében az alapértelmezés szerinti maximális intenzitás függ az aktuális pontszerű/reflektor tompulás beállításától. Ha nincs tompulás, ez az érték 1.00; inverz lineáris

tompulás esetén az érték a rajz maximális kiterjedése; inverz négyzetes tompulás esetén pedig a maximális kiterjedés négyzete.

R—Rename (Átnevez)

Átnevez egy fényforrást.

(c:fény "R" *old_name new_name*)

Az *old_name* argumentum egy karakterlánc, amely megadja az átnevezni kívánt fényforrást. A *new_name* argumentum olyan karakterlánc, amely a fényforrás új nevét adja meg .

```
(c:fény "R" "PONT5" "OLDALPT")
```

Külsőleg definiált függvény

render ARX alkalmazás

FORGAT3D

Egy objektumot forgat egy tetszőleges háromdimenziós tengely körül

(forgat3d *arg1 arg2 ...*)

A **forgat3d** függvény argumentumainak sorrendje, száma és típusa megegyezik azzal, amit a parancssorba billentyűzne be. Üres válasz megjelölésére (↵) használja a `nil` értéket vagy az üres karakterláncot (`""`). Sikeres hívás esetén a **forgat3d** függvény `T` értékkel tér vissza, egyébként `nil` a visszatérési érték.

A következő példa a *p1* és *p2* pontok által megadott tengely körül 30 fokkal forgatja el a kiválasztott objektumokat.

```
(setq ss (ssget))
(forgat3d ss p1 p2 30)
```

Megjegyzés Az AutoLISP a SAGET könyvtár használatával támogatja a **forgat3d** függvény működését.

Külsőleg definiált függvény

geom3d ADS alkalmazás

GIFBE

Egy GIF formátumú raszteres képfájlt importál a rajzba

(c:gifbe *filename*)

A **c:**gifbe függvény egy karakterlánc-argumentumot fogad, amely az importálni kívánt GIF fájl nevét adja meg.

Külsőleg definiált függvény

rasterin ADS alkalmazás

ILLESZT

Objektumokat mozgat és forgat, lehetővé téve más objektumokhoz való illesztésüket

(illeszt *arg1 arg2 ...*)

Az **illeszt** függvény argumentumainak sorrendje, száma és típusa megegyezik azzal, mint amit a parancssorba billentyűzne be. Üres válasz megjelölésére (↵) használja a `nil` értéket vagy az üres karakterláncot (`""`). Sikeres hívás esetén az **illeszt** függvény `T` értékkel tér vissza, egyébként `nil` a visszatérési érték.

A következő példa két pár forrás- és célpontot ad meg, így kétdimenziós mozgás jön létre.

```
(setq ss (ssget))
(illeszt ss s1 d1 s2 d2 "" "2d")
```

Megjegyzés Az AutoLISP a SAGET könyvtár használatával támogatja az **illeszt** függvény működését.

Külsőleg definiált függvény

geom3d ADS alkalmazás

JELENET

Új jelenetet hoz létre, illetve már meglévőt módosít vagy töröl (csak a papírtérben)

(c:jelenet *mode [options]*)

Ennek a függvénynek hat módja van. Minden módot egy karakterlánc ad meg. A további argumentumok a módtól függnek.

A JELENET függvény módjai

Mode	*Leírás*
D	Létező jelenetet töröl
L	A rajzon található összes jelenetet kilistázza vagy a megnevezett jelenetek adatait adja vissza
M	Létező jelenetet módosít
N	Új jelenetet hoz létre
R	Létező jelenetet nevez át
S	Beállítja az aktuális jelenetet

D—Delete (Töröl)

Létező jelenetet töröl.

(c:jelenet "D" *name*)

A *name* argumentum egy karakterlánc, amely megadja a törölni kívánt jelenet nevét. Amennyiben az aktuális jelenetet törölte, *NONE* lesz az aktuális jelenet.

```
(c:jelenet "D" "PLANVIEW")
```

L—List (Listáz)

A rajzon található összes jelenetet kilistázza, vagy a megnevezett jelenetek adatait adja vissza.

(c:jelenet "L" *[name]*)

A *name* argumentum egy karakterlánc, amely megadja a kilistázni kívánt jelenet nevét. Amennyiben a *name* argumentumot elhagyja, a **c:jelenet** függvény a rajzon definiált összes jelenet listájával tér vissza. Ha megadja a *name* argumentumot, a **c:jelenet** függvény a megnevezett jelenetek adataival tér vissza.

A következő programrészlet a rajzban definiált jelenetek neveit adja vissza.

```
(c:jelenet "L")                    eredménye:
("" "JELENET1" "JELENET2" "JELENET3")
```

Az üres karakterlánc ("") által megadott alapértelmezés szerinti jelenetet, melynek neve *NONE*, nem lehet módosítani.

```
(c:jelenet "L" "JELENET2")
```

a megnevezett jelenet adatait adja vissza.Következzék néhány példa:

```
(T T)                              Az *aktuális* nézet az *összes* fényforrással
("NÉZET1" nil)                     A nevezett nézet fényforrás nélkül
("NÉZET2" ("FÉNY1" "FÉNY2"))       A nevezett nézet két fényforrással
```

M—Modify (Módosít)

Létező jelenetet módosít.

(c:jelenet "M" *name [view [lights]]*)

A Modify mód opciói megegyeznek a New mód opcióival, azzal a különbséggel, hogy a *view* argumentumnak `nil` értéket is adhat, ha csak a fényforrást kívánja módosítani.

Megjegyzés A *lights* argumentumot listaként kell megadni, még akkor is, ha csak egy fényforrást kíván módosítani.

A következő függvényhívás például a FRONT nézet és a rajzon található összes fényforrás segítségével módosítja a JELENET1 nevű jelenetet :

```
(c:jelenet "M" "JELENET1" "FRONT" (C:FÉNY "L"))
```

Az alábbi függvényhívás a BACK nézet és a rajzon található P1 illetve P2 nevű fényforrások segítségével módosítja JELENET1 nevű jelenetet :

```
(c:jelenet "M" "JELENET1" "BACK" '("P1" "P2"))
```

N—New (Új)

Új jelenetet hoz létre.

(c:jelenet "N" *name [view [lights]]*)

A New mód argumentumait az alábbi táblázat tartalmazza.

A JELENET függvény "N" módjának argumentuma

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
name	STR	Új jelenet neve	Nincs
view	STR	Egy AutoCAD által elnevezett nézet neve	Nincs
	T (SYM)	*CURRENT* nézet	
lights	LIST (karakterláncok)	Az összes fényforrás nevének listája	A rajz összes fényforrása, *ALL*
	T (SYM)	Az *ALL* (összes) fényforrást használja a rajzon	
	nil	Nem használ fényforrást a rajzban	A néző "válla felett" elhelyezett távoli fényforrás

Megjegyzés A `lights` argumentumot listaként kell megadni, még akkor is, ha csak egy fényforrást kíván módosítani.

A következő példa a *CURRENT* nézetet és az *ALL* fényforrást használva hoz létre egy DEFAULT nevű jelenetet :

```
(c:jelenet "N" "DEFAULT")
```

Az alábbi példa a *CURRENT* nézet és a "váll feletti" fényforrás segítségével hozza létre a MATT nevű jelenetet:

```
(c:jelenet "N" "MATT" T nil)
```

Az alábbi függvényhívás a SAJÁT_NÉZET nevű nézet és a NAP, LÁMPA és REFLEKTOR fényforrások felhasználásával hozza létre a KÜLÖNLEGES nevű jelenetet:

```
(c:jelenet "N" "KÜLÖNLEGES" "SAJÁT_NÉZET" '("NAP" "LÁMPA
"REFLEKTOR"))
```

R—Rename (Átnevez)

Átnevez egy jelenetet.

(c:jelenet "R" *old_name new_name*)

Az *old_name* argumentum az eredeti jelenet nevét tartalmazó karakterlánc, a *new_name* pedig a jelenet új nevét adja meg.

Például:

```
(c:jelenet "R" "KÜLÖNLEGES" "FÉNYES")
```

S—Set (Beállít)

Beállítja az aktuális jelenetet.

(c:jelenet "S" *[name]*)

A *name* argumentum egy karakterlánc, amely megnevezi az aktualizálni kívánt jelenetet. Amennyiben a *name* elmarad, a **c:jelenet** függvény az éppen kiválasztott jelenet nevével tér vissza. Amennyiben nincs aktuális jelenet és a *name* argumentum elmarad, a **c:jelenet** függvény egy üres karakterlánccal (`""`) tér vissza.

Az alábbi függvényhívás a JELENET3 nevű jelenetet jelöli ki aktuális jelenetnek:

```
(c:jelenet "S" "JELENET3")
```

Külsőleg definiált függvény

render ARX alkalmazás

KAL

Meghívja az on-line geometriai számológépet, és a kiértékelt kifejezés értékével tér vissza

(kal *expression*)

Az *expression* argumentum egy idézőjelek közötti karakterlánc. A "'KAL" függvényről további információ található az AutoCAD on-line súgójában.

A következő példa a KAL parancsot egy AutoLISP kifejezésen belül használja a **trans** függvénnyel együtt:

```
(trans (kal "[1,2,3]+KÖZ") 1 2)
```

Külsőleg definiált függvény

geomcal ADS alkalmazás

KÉPMENTŐ

Egy fotorealisztikus képet TGA, TIFF vagy GIF formátumú fájlba ment el

(c:képmentő *filename type [portion] [xoff yoff xsize ysize] [compression]*)

Amennyiben az AutoCAD számára egy külön fotorealisztikus megjelenítőt konfiguráltak, a `portion` argumentumot nem szabad megadni. Megadhatja a kép méretét és eltolását, és a TGA illetve TIFF fájlok számára a tömörítés adatait. Az argumentumokat az alábbi táblázat tartalmazza.

A KÉPMENTŐ függvény argumentumai

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
filename	STR	Képfájl-név	Nincs
type	STR	Fájltípusok: TGA, TIFF vagy GIF	Nincs
portion	STR	Az elmentésre kijelölt képernyőterület: A—aktív nézetablak D—rajzterület F— teljes képernyő	"A"
xoff	INT	*X* irányú eltolás pixelben	0
yoff	INT	*Y* irányú eltolás pixelben	0
xsize	INT	*X* irányú méret pixelben	Tényleges *X* irányú méret
ysize	INT	*Y* irányú méret pixelben	Tényleges *Y* irányú méret
compression	STR	Tömörítés adatai: NONE PACK (csak TIFF fájlok) LZW (csak TIFF fájlok) RLE (csak TGA fájlok)	Nincs

Megjegyzés Ha GIF típusú fájl van megadva a függvény figyelmen kívül hagyja a `compression` argumentumot.

Az alábbi példa a teljes képernyő tartalmát menti el *teszt.tga* néven egy TIFF formátumú fájlba, LZW tömörítést használva:

```
(c:képmentő "TESZT" "TIF" "LZW")
```

Külsőleg definiált függvény

render ARX alkalmazás

KÉPÚJRA

Egy GIF, TGA vagy TIFF képet jelenít meg

`(c:képújra filename type [xoff yoff xsize ysize])`

A KÉPÚJRA paranccsal egy GIF, TGA vagy TIFF formátumú fájl tartalmát nézheti meg az AutoCAD fotorealisztikus megjelenítő eszközén. Használja ezt a függvényt a képfájlok különböző eltolású és méretű megjelenítésére. A függvény argumentumait a következő táblázatban írtuk le:

A KÉPÚJRA függvény argumentumai

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
filename	STR	Képfájl-név	Nincs
type	STR	Érvényes értékek: GIF, TGA vagy TIFF	Nincs
xoff	INT	A kép *X* irányú eltolása pixelben	0
yoff	INT	A kép *Y* irányú eltolása pixelben	0
xsize	INT	A kép *X* mérete pixelben	Tényleges *X* méret
ysize	INT	A kép *Y* mérete pixelben	Tényleges *Y* méret

A következő függvényhívás például eltolás nélkül olvas be egy *teszt.tga* nevű,500 × 400 pixel méretű képet :

```
(c:képújra "TESZT" "TGA" 0 0 500 400)
```

Külsőleg definiált függvény

render ARX alkalmazás

PCXBE

Egy PCX formátumú raszteres képfájlt importál a rajzba

(c:pcxbe *filename*)

A **c:pcxbe** függvény argumentuma egy karakterlánc, amely megadja az importálni kívánt GIF fájlt.

Külsőleg definiált függvény

rasterin ADS alkalmazás

PSBE

Egy PostScript fájlt importál

(c:psbe *filename position scale*)

A *filename* argumentum a PostScript kép nevét tartalmazó karakterlánc. Az *.eps* fájlkiterjesztést nem kell megadni. A *position* argumentum a (névtelen) PostScript blokk beillesztési pontját megadó pont. A *scale* argumentum a méretarány-tényezőt megadó valós érték. Sikeres hívás esetén a **c:psbe** függvény az újonnan létrehozott objektum nevét adja vissza, hiba esetén pedig `nil` a visszatérési érték.

A következő függvényhívás importálja a *minta.eps* nevű PostScript fájlt, és azt 25-ös méretarány-tényezővel illeszti be a (24,19) pontra:

```
(c:psbe "minta" '(24 19) 25)
```

A PostScript kitöltései a rajzfájlokban bővített rajzelemadatokként tárolódnak, és az `AUTOCAD_POSTSCRIPT_FIGURE` alkalmazásnév azonosítja őket.

Külsőleg definiált függvény

acadps ADS alkalmazás

PSKÉP

Egy importált PostScript kép megjelenítését vezérli, miközben azt a PSBE parancs adott helyre vontatja

(c:pskép *mode*)

A *mode* argumentum egy egész szám, melynek értéke 0 vagy 1. A PSKÉP aktuális értéke a PSBE parancs interaktív használatát befolyásolja. Amennyiben PSKÉP értéke 1, a PSBE parancs folyamatosan kirajzolja a PostScript képet, miközben a felhasználó vontatással változtatja annak léptékét. Ha a PSKÉP értéke 0, a PSBE parancs csak a kép befoglaló keretét rajzolja ki vontatás közben. Sikeres hívás esetén a **c:pskép** függvény a PSKÉP régi értékével, hiba esetén pedig `nil` értékkel tér vissza.

A következő függvényhívás az 1 érték beállításával bekapcsolja a PSKÉP módot. Ezt követően a PSBE interaktív használatakor folyamatosan kirajzolja a PostScript képet, miközben a felhasználó vontatással változtatja annak léptékét.

```
(c:pskép 1)
```

Külsőleg definiált függvény

acadps ADS alkalmazás

PSKITÖLT

PostScript mintával tölt ki egy kétdimenziós vonalláncot

(c:pskitölt *ent pattern [arg1 [arg2]] ...*)

Az *ent* argumentum a vonallánc neve. A *pattern* argumentum a kitöltési minta nevét tartalmazó karakterlánc. A *pattern* karakterláncnak meg kell egyeznie az aktuális *acad.psf* fájlban definiált kitöltési minták egyikével. Az *arguments* argumentumok a belső PostScript kitöltési eljárás argumentumai: számuk és típusuk a *pattern* által igényelt argumentumoknak felel meg, az *acad.psf* fájlban szereplő definíció szerint. Minden argumentum egész vagy valós számérték. Számuk mintánként 0-tól 25-ig terjedhet. Ha a függvényhívásban kevesebb argumentum szerepel, mint amennyit a minta definíciója megkövetel, akkor a fennmaradó argumentumok helyett az AutoCAD a minta alapértelmezés szerinti értékeit veszi figyelembe. Sikeres hívás esetén a **c:pskitölt** függvény `T`, hiba esetén pedig `nil` értékkel tér vissza.

A Grayscale kitöltési mintának egyetlen argumentuma van. A következő függvényhívás az alapértelmezés szerinti Grayscale argumentumot használja fel, melynek értéke 50 százalék:

```
(c:pskitölt ename "Grayscale")
```

Ez a hívás pedig 10 százalékos szürke árnyalatot határoz meg:

```
(c:pskitölt ename "Grayscale" 10)
```

A PostScript kitöltései a rajzfájlokban bővített rajzelemadatokként tárolódnak, és az AUTOCAD_POSTSCRIPT_FIGURE alkalmazásnév azonosítja őket.

Külsőleg definiált függvény

acadps ADS alkalmazás

RANY

Render anyagokat hoz létre; szerkeszti, beilleszti és leválasztja azokat

(c:rany *mode options*)

Ennek a függvénynek hat módja van, melyek az alábbi táblázatban láthatók. Minden módot egy karakterlánc-argumentum ad meg.

Az RANY függvény módjai

Mode	*Leírás*
A	Anyagot rendel az objektumhoz
C	Anyagot másol
D	Anyagot választ le
L	A rajz összes anyagát kilistázza, vagy a megadott anyag adatait adja vissza
M	Anyagot módosít
N	Új anyag

A—Attach (Hozzárendel)

Anyagot rendel a kiválasztott objektumokhoz, vagy egy AutoCAD Szín Index (AutoCAD Color Index, ACI) értékhez.

(c:rany "A" *name [aci_selset]*)

A *name* argumentum egy karakterlánc, amely megadja a rendelni kívánt anyag nevét. Az opcionális *aci_selset* argumentum azt az ACI (egész) számot vagy kiválasztási halmazt adja meg, amelyhez az anyagot rendeli.

Például:

```
(c:rany "A" "LILA TIGRIS" 1)
```

a LILA TIGRIS nevű anyagot rendeli az ACI 1-es (vörös) értékéhez.

Amennyiben elhagyja az *aci_selset* argumentumot, az Attach mód két tétel listájával tér vissza. Az első az anyag egész típusú indexe, a második pedig azt a kiválasztási halmazt tartalmazza, amelyhez az anyagot rendelte. Például:

Parancs: **(c:rany "a" "twood")**
Objektumok gyűjtése...1 talált
(258 <Kiválasztási halmaz: 10>))

A 0 és 256 közötti anyagindex érték egy ACI szám; a 256-nál nagyobb index azt mutatja, hogy az AutoCAD Render anyagot nem az ACI jelölte ki.

C—Copy (Másol)

Új anyagot hoz létre azáltal, hogy a rajzba egy már meglévőt másol.

(c:rany "C" *cur_name new_name*)

A *cur_name* argumentum egy karakterlánc, amely megadja a másolni kívánt anyagnevet. A *new_name* argumentum az új anyag nevét megadó karakterlánc. Ezek után adatainak megváltoztatásával módosíthatja az új (vagy régi) anyagot.

```
(c:rany "C" "VÖRÖS" "VÖRÖS2")
```

D—Detach (Leválaszt)

Anyagot választ le a kiválasztási halmazból vagy egy ACI értékből.

(c:rany "D" *[aci_selset]*)

Az opcionális *aci_selset* argumentum megadja az ACI számot (mely egész típusú) vagy a kiválasztási halmazt, amelyből leválasztja az anyagot. Például:

```
(c:rany "D" (ssget))
```

arra kéri a felhasználót, hogy válasszon ki objektumokat, azután minden objektumról leválasztja annak anyagát.

L—List (Listáz)

A rajzon található anyagok adatait listázza ki.

```
(c:rany "L" [name])
```

Az opcionális *name* argumentum egy karakterlánc, amely a listázni kívánt anyag adatait adja meg. Amennyiben elhagyja a *name* argumentumot, **c:rany** függvény kilistázza a rajzon található összes anyagot. Például:

Parancs: **(c:rany "L")**
("*GLOBAL*" "KÉK ÜVEG" "FEHÉR PLASZTIK" "TFA" "BEIGE MATTE")

A lista első karakterlánca megadja az alapértelmezés szerinti globális anyagot, *GLOBAL*. Ezt ugyanúgy adhatja át a `(c:rany)` függvénynek, mintha könyvtárneveket, vagy felhasználó által definiált anyagokat adna át:

Parancs: **(c:rany "L" "*GLOBAL*")**
("*GLOBAL*" (-1.0 -1.0 -1.0) 1.0 ("" 0.0 0 (1.0 1.0) (0.0 0.0) 0.0) (0.0 0.0 0.0) 1.0 (1.0 1.0 1.0) 1.0 ("" 0.0 0) 0.5 0.0 ("" 0.0 0 (1.0 1.0) (0.0 0.0) 0.0) 1.0 ("" 0.0 0 (1.0 1.0) (0.0 0.0) 0.0))

Az anyag adataiban szereplő listaelemek megegyeznek a Modify vagy a New módok argumentumaival.

M—Modify (Módosít)

Meglévő anyagot módosít.

```
(c:rany "M" name [color [color-wgt [ambient [amb-wgt
[refl [refl-wgt [roughness ]]]]]]])
```

A Modify mód opciói megegyeznek a New mód opcióival. Ha egy argumentum értéke `nil`—vagy elmarad a lista végéről—, akkor megőrzi aktuális értékét.

N—New (Új)

Új anyagot hoz létre.

```
(c:rany "N" name [color [color-wgt [ambient [amb-wgt
[refl [refl-wgt [roughness ]]]]]]])
```

A New mód argumentumait a következő táblázat tartalmazza.

Az RANY függvény "N" módjának argumentumai

Argumentum	*Adattípus*	*Leírás*	*Alapértelmezés*
name	STR	A létrehozni kívánt anyag neve	Nincs
color	LIST (valós számok)	Anyagszín RGB színhármasokban; a (–1.0 –1.0 –1.0) színhármas azt jelenti, hogy a szín egy objektum ACI értékéből származik	(–1.0 –1.0 –1.0)— ACI szerint
color-wgt	REAL	Súly lépték (szín értéke) —a szórt fény nagysága	1.0
ambient	LIST (valós számok)	Árnyék színe, mint RGB színhármas	(0.0 0.0 0.0)—fekete
amb-wgt	REAL	Súly tényező (háttér értéke)—a háttérszín nagysága	1.0
refl	LIST (of reals)	Tükröződés (specular) színe, mint RGB színhármas	(1.0 1.0 1.0)—fehér
refl-wgt	REAL	Súly tényező (tükröződés értéke)—a tükröző fény nagysága	1.0
roughness	REAL	Érdesség—a tükröződő terület mérete	0.50

Külsőleg definiált függvény

render ARX alkalmazás

RBEÁLL

Beállítja a fotorealisztikus megjelenítés paramétereit

(c:rbeáll *mode option [setting]*)

Az RBEÁLL meghatározza, hogy melyik megjelenítési paramétereket használja, és melyik megjelenítési viselkedés lesz az alapértelmezés . A függvénynek hat módja van; ezeket az alábbi táblázat tartalmazza. Minden módot egy karakterlánc ad meg. A további argumentumok a módtól függnek.

Az RBEÁLL függvény módjai

Mode	*Leírás*
ACMAP	Paletta opció a nézetablak megjelenítéshez
DEST	Kimenet
ICON	A Fény és Anyagok ikonblokkok mérete
SELECT	Legyen-e prompt az objektum kiválasztására
STYPE	Megjelenítés típusa
TOGGLE	Megjelenítési opciók

ACMAP—AutoCAD paletta

Kiválasztja, hogy melyik paletta-opciót használja a nézetablak megjelenítéshez.

(c:rbeáll "ACMAP" *option*)

Az *option* argumentum a paletta-opciót megadó karakterlánc. Az *option* lehetséges értékeit a következő táblázat tartalmazza:

Az RBEÁLL függvény "ACMAP" opciója

Érték	*Leírás*
ACAD	Állandó AutoCAD paletta
FOLD	Optimális kiosztás/Leképzés nélkül
NOFOLD	Optimális kiosztás/Leképzés

A következő függvényhívás például megadja a leképzés opciót:

```
(c:rbeáll "ACMAP" "FOLD")
```

DEST—Kimenet

Kiválasztja a használt kimeneti anyagot.

(c:rbeáll "DEST" *option*)

Az *option* argumentum egy karakterlánc, amely megadja a fotorealisztikus megjelenítés kimenetét. Az *option* lehetséges értékeit a következő táblázat tartalmazza.

Az RBEÁLL függvény "DEST" opciója

Érték	*Leírás*
FRAMEBUFFER	A valósághű kép a megjelenítő anyagra
HARDCOPY	A valósághű kép fájlba vagy a papírmásolat készítő anyagra

A következő függvényhívás például a valósághű képet egy fájlba írja (feltéve, hogy a *rhexport* meghajtó konfigurálva van):

```
(c:rbeáll "DEST" "HARDCOPY")
```

ICON—Ikon

Megadja a fényforrás vagy anyag ikonblokk méretét a rajzon.

(c:rbeáll "ICON" *option*)

Az *option* argumentum egy valós szám, amely megadja az ikonblokk méretét (az alapértelmezés szerinti érték 1.00). A következő függvényhívás 50 százalékra változtatja az ikon méretét:

```
(c:rbeáll "ICON" 0.5)
```

STYPE—Rendertípus

Megadja a használt megjelenítési típust.

(c:rbeáll "STYPE" *option*)

Az *option* argumentum a megjelenítés típusát megadó karakterlánc.

A következő függvényhívás például meghatározza, hogy a következő valósághű kép megjelenítését az AutoCAD megjelenítő fogja elvégezni.

```
(c:rbeáll "STYPE" "CRENDER")
```

SELECT—Kiválaszt

Megadja, hogy legyen-e prompt az objektum kiválasztására még a valósághű kép megjelenítése előtt.

(c:rbeáll "SELECT" *option*)

Az *option* argumentum egy karakterlánc, amely megadja a promptot. Az *option* lehetséges értékeit a következő táblázat tartalmazza.

Az RBEÁLL függvény "SELECT" opciója

Érték	*Leírás*
ALL	A teljes ábrát jeleníti meg
ASK	Objektumkiválasztást kér

Ez a függvényhívás objektumkiválasztást kér a felhasználótól:

```
(c:rbeáll "SELECT" "ASK")
```

TOGGLE—Kapcsoló

Különböző megjelenítési opciókat vezérel.

(c:rbeáll "TOGGLE" *option setting*)

Az *option* argumentum egy karakterlánc, amely magadja a promptot. Az *option* lehetséges értékeit a következő táblázat tartalmazza. A *setting* argumentum egy karakterlánc, amely megadja a kapcsoló állapotát. A *setting* lehetséges értékei "ON" és "OFF".

Az RBEÁLL függvény "TOGGLE" opciója

Érték	*Leírás*
SMOOTH	Finomítás nélküli megjelenítés
MERGE	Háttérre másolja az objektumokat
FINISH	Anyagokat alkalmaz
SKIPRDLG	Nem jeleníti meg a Render párbeszédpanelt

A következő függvényhívások közül az első kikapcsolja a Merge opciót, a második pedig bekapcsolja az árnyékot:

```
(c:rbeáll "TOGGLE" "MERGE" "OFF")
(c:rbeáll "TOGGLE" "SMOOTH" "ON")
```

Külsőleg definiált függvény

render ARX alkalmazás

RENDER

A geometriai, megvilágítási és a felület minőségére vonatkozó információk alapján elkészíti a háromdimenziós drótváz modell valósághűen árnyalt képét

(c:render *[filename]*)

Ha a *filename* argumentum jelen van, a függvény a fotorealisztikus megjelenítést az argumentum által megadott fájlba írja. Ha nem konfigurálja a fotorealisztikus megjelenítés meghajtóját, a függvény figyelmen kívül hagyja a *filename* argumentumot. Az aktuális konfigurációnak egy fájlt kell megadnia a fotorealisztikus megjelenítés célpontjául. A megjelenítést az aktuális beállítások vezérlik; ezek beállítására használja a **c:rbeáll** függvényt.

Megjegyzés Amennyiben az aktuális render beállításokban a Válasszon objektumokat beviteli mód szerepel, és a PICKFIRST rendszerváltozó be van kapcsolva, valamint ha a **c:render** függvény meghívásakor van aktuális kiválasztási halmaz , akkor további prompt megjelenése nélkül kerül sor a halmaz objektumainak fotorealisztikus megjelenítésére.

Az AutoCAD Render konfigurálásáról további információ található az *AutoCAD Telepítési útmutató* 6., "AutoCAD Render" című fejezetében.

Külsőleg definiált függvény

render ARX alkalmazás

RENDERKÉP

Újra megjeleníti a RENDER parancs által utoljára létrehozott fotorealisztikus képet

(c:renderkép)

A RENDERKÉP parancsot az ablakkezelést nem támogató, teljes képernyős megjelenítésre konfigurált, egyképernyős megjelenítővel

rendelkező rendszerek esetében használhatja. Ebben az esetben a kép az egész képernyőt betölti. A grafikus és szöveges képernyő között az F2 billentyűvel lehet váltogatni.

Külsőleg definiált függvény

render ARX alkalmazás

RIASPECT

Megváltoztatja a kép szélesség-magasság arányát a GIFBE, PCXBE és TIFFBE parancsoknál

```
(c:riaspect value)
```

A `value` argumentum egy valós szám.

További tudnivalók

A RIASPECT rendszerváltozóról az AutoCad on-line súgójában található további információ.

Külsőleg definiált függvény

rasterin ADS alkalmazás

RIBACKG

Megadja a háttér színét a GIFBE, PCXBE és TIFFBE parancsoknál

```
(c:ribackg value)
```

A `value` argumentum egy egész szám.

További tudnivalók

A RIBAKCG rendszerváltozóról az AutoCad on-line súgójában található további információ.

Külsőleg definiált függvény

rasterin ADS alkalmazás

RIEDGE

Az élvizsgálatot vezérli a GIFBE, PCXBE és TIFFBE parancsoknál

```
(c:riedge value)
```

A `value` argumentum egy egész szám.

További tudnivalók

A RIEDGE rendszerváltozóról az AutoCad on-line súgójában található további információ.

Külsőleg definiált függvény

rasterin ADS alkalmazás

RIGAMUT

Az importált kép szineinek számát vezérli a GIFBE, PCXBE és TIFFBE parancsoknál

```
(c:rigamut value)
```

A `value` argumentum egy egész szám.

További tudnivalók

A RIGAMUT rendszerváltozóról az AutoCad on-line súgójában található további információ.

Külsőleg definiált függvény

rasterin ADS alkalmazás

RIGREY

Egy kép szürkeárnyalatban történő importálását vezérli a GIFBE, PCXBE és TIFFBE parancsoknál

```
(c:rigrey value)
```

A `value` argumentum egy egész szám.

További tudnivalók

A RIGREY rendszerváltozóról az AutoCad on-line súgójában található további információ.

Külsőleg definiált függvény

rasterin ADS alkalmazás

RITHRESH

Egy kép világosságon alapuló importálását vezérli a GIFBE, PCXBE és TIFFBE parancsoknál

`(c:rithresh value)`

A `value` argumentum egy egész szám.

További tudnivalók

A RITHRESH rendszerváltozóról az AutoCad on-line súgójában található további információ.

Külsőleg definiált függvény

rasterin ADS alkalmazás

RKONFIG

Újrakonfigurálja a fotorealisztikus megjelenítéssel kapcsolatos beállítást

`(c:rkonfig)`

Az AutoCAD először közli a rendszer aktuális konfigurációját, majd a konfigurációval kapcsolatos információt kér. Mivel az RKONFIG parancs interaktív, nem szokás egy alkalmazásból meghívni (Szintaktikailag ugyanakkor beépíthető).

A megjelenített választási lehetőségek a használt operációs rendszertől függenek. Az *AutoCAD Telepítési útmutató* 6., "AutoCAD Render" című fejezete tárgyalja az elérhető render opciókat.

Külsőleg definiált függvény

render ARX alkalmazás

RSTAT

Statisztikát állít össze az utolsó fotorealisztikus megjelenítésről

`(c:rstat [filename])`

Az RSTAT parancs információt szolgáltat az utolsó fotorealisztikus megjelenítésről.

Amennyiben fájlnevet is megad, nem jelenik meg párbeszédpanel, és az információt az adott fájlba menti el. Ha nem ad meg fájlnevet, az AutoCAD a Statisztika nevű párbeszédpanelben jeleníti meg a fotorealisztikus megjelenítéssel kapcsolatos információt .

A következő példa a *alakok.txt* nevű fájlba írja az utolsó fotorealisztikus megjelenítéssel kapcsolatos információkat:

```
(c:rstat "alakok.txt")
```

Külsőleg definiált függvény

render ARX alkalmazás

TIFFBE

Egy TIFF formátumú raszteres képfájlt importál a rajzba

(c:tiffbe *filename*)

A **c:tiffbe** függvény egy karakterlánc-argumentumot fogad, amely magadja az importálni kívánt TIFF fájlt.

Külsőleg definiált függvény

rasterin ADS alkalmazás

TÜKRÖZ3D

Kiválasztott objektumokat tükröz a felhasználó által megadott síkra

(tükröz3d *arg1 arg2 ...*)

A **tükröz3d** függvény argumentumainak sorrendje, száma és típusa megegyezik azzal, mint amit a parancssorba billentyűzne be. Üres válasz megjelölésére (↵) használja a `nil` értéket vagy az üres karakterláncot (`""`). Sikeres hívás esetén a **tükröz3d** függvény `T` értékkel tér vissza, egyébként `nil` a visszatérési érték.

A következő példa a kiválasztott objektumokat a 0,0,5 ponton átmenő *XY* síkra tükrözi, majd törli a régi objektumokat.

```
(setq ss (ssget))
(tükröz3d ss "XY" '(0 0 5) "Y")
```

Megjegyzés Az AutoLISP a SAGET könyvtár használatával támogatja a **tükröz3d** függvény működését.

Külsőleg definiált függvény

geom3d ADS alkalmazás

VISLKONV

Visual Link™ adatokat konvertál át objektumokká az AutoCAD Render fotorealisztikus megjelenítése számára

(c:vislkonv *preserve*)

A *preserve* argumentum egy egész szám, amely megadja, hogy megőrzendők-e a létező AutoCAD Render anyag hozzárendelések. A **c:vislkonv** függvény megőrzi a hozzárendeléseket, ha a *preserve* értéke 0, és felülírja azokat, ha a *preserve* értéke 1.

A **c:vislkonv** függvény nem változtatja meg a Visual Link adatokat, így azokat továbbra is használhatja a Visual Link szoftverrel.

Az alábbi függvényhívás Visual Link adatokat konvertál anélkül, hogy felülírná az AutoCAD Render anyag hozzárendeléseket:

```
(c:vislkonv 0)
```

Külsőleg definiált függvény

render ARX alkalmazás

15

MEMÓRIAKEZELÉS

Az AutoCAD program képes arra, hogy a szükség szerint változtassa a memória felhasználását. A korlátozott memóriájú rendszerek esetében azonban szükség lehet további beavatkozásra is. Jelen fejezet az ilyen rendszerek kezelését tárgyalja.

Az ebben a kézikönyvben ismertetett összes szimbólumot, felhasználó által definiált függvényt, illetve szabvány függvényt az AutoCAD csak a rajzszerkesztés idején tárolja a memóriában. Az AutoLISP elindulásakor két nagy memóriaterületet foglal le. Az első, halomnak (*heap*) nevezett területen tárolja az összes függvényt és szimbólumot (melyeket *node*-nak, azaz csomópontnak is nevezünk); minél több szimbólumot és függvényt használ (és minél összetettebbek a függvények), annál nagyobb területet igényelnek majd. A második, veremnek (*stack*) nevezett terület a függvények argumentumainak és részeredményeinek tárolására szolgál. Minél mélyebben ágyazza egymásba a függvényeket, vagy minél több rekurzív hívást alkalmaz, annál nagyobb veremterületre van szükség.

Csomóponti terület

A csomópont (*node*) olyan memóriastruktúra, amely az AutoLISP összes adattípusát képes megjeleníteni. Jelenleg az AutoLISP 12 bájtos csomópontokat használ. A memória feldarabolását és a halom kezelésének nehézségeit elkerülendő, a csomópontok a halom területen *szegmenseknek* nevezett csoportokban helyezkednek el. Alapértelmezés szerint egy szegmens 514 csomópontból áll (6168 bájt).

Az AutoLISP egy listában nyilvántartja a szabad csomópontokat (azokat, amelyekhez éppen nincs szimbólum rendelve). Amikor egy szimbólum vagy egy érték tárolásához egy csomópontra van szükség, az AutoLISP a szabad csomópontok listájában kikeres egy felhasználható csomópontot. Amennyiben egyet sem talál, automatikusan elindít egy hulladékgyűjtő folyamatot, amelynek során a szimbólumoktól már megszabadított csomópontokat felveszi a szabad listára. Ezután, a kérésnek eleget téve, kiválaszt egyet.

Ha a hulladékgyűjtés nem eredményezett elegendő szabad csomópontot, az AutoLISP egy további szegmenst foglal le a halomterületből. Amennyiben ez sikerül, az új csomópontok a szabad listára kerülnek, és az AutoLISP ezekből választ ki egyet az eredeti kérés teljesítésére. Ha a halomterületből nem lehet lefoglalni további szegmenseket, az AutoLISP a virtuális függvénylapozás segítségével -- a legrégebben használt függvények kihelyezésével -- szabadít fel további csomópontokat; máskülönben az elégtelen node terület üzenetet jeleníti meg, és a csomóponti területet igénylő függvényt megszakítja. Fontos megemlíteni, hogy a csomóponti terület *nem* kerül vissza a halomterülethez, amíg Ön ki nem lép az AutoCAD programból .

A hulladékgyűjtő folyamatot a **gc** függvény segítségével hajthatja végre:

```
(gc)
```

A hulladékgyűjtés azonban időigényes művelet, ezért ne alkalmazza feleslegesen. Legjobb azt az AutoLISP automatikus mechanizmusára bízni, amely csak a feltétlenül szükséges esetekben végzi el a műveletet.

Csomóponti terület felszabadítása

Amennyiben olyan függvényeket és szimbólumokat hoz létre, amelyekre csak rövid ideig van szüksége, és már nem veszi hasznukat, *törölje* őket úgy, hogy *nil* értéket rendel hozzájuk. Például, ha a **setup** függvényt már betöltötte és használta, és a továbbiakban nincs szüksége rá, akkor a

```
(setq setup nil)
```

értékadással szabadulhat meg tőle. A *nil* érték hozzárendelésekor a függvény által használt csomóponti terület felszabadul, és más függvények vagy szimbólumok rendelkezésére áll.

Amennyiben egy FILE típusú (az **open** függvény által létrehozott) szimbólum által használt csomóponti területet kíván felszabadítani, le kell zárnia a fájlt, mielőtt nil értéket rendelne hozzá. Ha nem sikerül lezárni egy fájlt, az általa használt terület továbbra is foglalt marad, ezáltal csökken az újonnan megnyitható fájlok száma.

Megjegyzés A Release 12 verziót megelőzően az AutoCAD az `atomlist` szimbólumhoz rendelt listában tartotta nyilván az összes definiált függvényt és szimbólumot. Ez a lista már nem létezik ebben a formában; ennélfogva az "atomlista darabolásának" módszere sem alkalmazható többé a függvények és szimbólumok kitörlésére. Továbbra is megkaphatja azonban a definiált függvények és szimbólumok listáját, mégpedig az **atoms-family** függvény segítségével:

```
(atoms-family 0)
```

Ez a függvény a teljes lista, vagy akár egyes kiválasztott szimbólumok lekérdezésére is alkalmas. További információt talál az **"atoms-family"** függvénynél, a 13. fejezetben.

Technikai megjegyzések

Az alábbi információk tapasztalt LISP-programozóknak szólnak. Olyan belső AutoLISP mechanizmusokat ismertetnek, amelyek külön értesítés nélkül megváltozhatnak.

Karakterlánc-terület

A karakterláncokat tároló memóriaterület ugyanzon a halomterületen helyezkedik el, mint a csomóponti szegmensek. Ha a programja a későbbiekben ismertetett manuális allokációs függvények segítségével az összes hozzáférhető memóriát a csomópontok számára foglalja le, akkor nagy valószínűséggel az elégtelen karakterlánc-terület hibaüzenetet fogja kapni. Ajánlatos ennek kezelését a csomópontok automatikus allokációjára hagyni. A karakterlánc-terület a szimbólumnevektől kezdve a promptokon át a menük karakterláncá-ig, az AutoLISP kiértékelőjének átadott összes karakterlánc tárolására szolgál. Amennyiben kiértékelendő AutoLISP kifejezéseket tartalmazó hosszú, menüpontokat használ, ezek jelentős összefüggő karakterlánc területet igényelnek a halomból.

Szimbólumok tárolása

Az AutoLISP memóriastruktúrája mutatókon (pointereken) alapszik. A csomópontok használata meghatározó, és minden csomópontstruktúraként jelenik meg. Egy olyan egyszerű művelet, mint egy érték

szimbólumhoz rendelése, két csomópontot igényel: egyet a szimbólumnév, és egyet az érték tárolására.

```
(setq longsym 3.1415)
```

Ha a szimbólum neve hat, vagy ennél kevesebb karakterből áll, a név közvetlenül a szimbólumnév csomópontjában tárolódik; máskülönben a név tárolása a halomterületen kijelölt karakterlánc-területen történik, és a szimbólumnév csomópontja erre a karakterláncra mutat. Azaz, ha rövid (hat vagy ennél kevesebb karakterből álló) szimbólumneveket használ , csökken a karakterlánc-terület számára szükséges hely, és a halomterület kevésbé darabolódik fel.

Manuális allokáció

Az **alloc** és az **expand** függvények segítségével saját maga gyakorolhat ellenőrzést a csomóponti és a karakterlánc-terület kiosztása felett. Ha ezeket a kifejezéseket az *acad.lsp* fájl elején használja, előzetesen kijelölheti a csomópontokat, és lefoglalhat karakterlánc-területet is. Ezáltal ritkábban kell a szabad területek összegyűjtéséhez folyamodnia, azaz csökken az alkalmazás futási ideje.

Az **alloc** függvényt arra is használhatja, hogy a későbbiekben használt szegmensek méretét 514 csomóponttól eltérő értékre állítsa.

```
(alloc szám)
```

Az **alloc** függvény a szegmens méretét *szám* csomópontra állítja be, és a megelőző értékkel tér vissza.

Az **expand** függvénnyel saját maga jelölhet ki csomópontokat, a lefoglalni kívánt szegmensek számának megadásával.

```
(expand szám)
```

A *szám* argumentum a lefoglalni kívánt szegmensek számát jelöli.

Az **expand** függvény a halomterületen ténylegesen kijelölt szegmensek számával tér vissza, amely a fennmaradó halomterület nagyságától függően kevesebb is lehet az Ön által megadott értéknél.

Példa

```
(alloc 1024)
```

1024 csomópontra állítja a szegmensméretet; mindegyik szegmens esetében 12488 bájt memóriát foglal le a halomterületből.

```
(alloc 100)
```

a szegmensméretet 100-ra állítja; ez csak 1200 bájt memóriát igényel szegmensenként.

Ha a szegmens mérete az alapértelmezés szerinti 512 csomópontra van állítva, a következő függvényhívás 10 szegmenst (10x12x512=61480 bájtot) foglal le.

```
(expand 10)
```

Több szegmenst is lefoglalhat, mint amennyi rendelkezésre áll.

Következzék itt egy gyakorlati példa:

```
(alloc 3000)      A szegmens méretét 3000 csomópontra állítja be
(expand 1)        Vesz 3000 szabad csomópontot (egy szegmens)
(alloc 10000)     További szegmensek hozzáadásának elkerülésére nagy
                  szegmensméretet állít be
```

A fenti programrészlet a halomterületből 36000 bájtot használ fel a csomópontokhoz, a maradékot a karakterlánc-terület számára tartja fenn. A csomópontokat előzetesen lefoglalja, és elhelyezi a szabad listán. Amíg fel nem használja mind a 3000 csomópontot, nem lesz szükség a maradék terület (hulladék) összegyűjtésére. Amint az összes csomópontot felhasználta, a további helyigény kielégítéséhez szükség lesz a takarításra is. Az `(alloc 10000)` függvényhívás olyan értékre állítja be a szegmensméretet, amely lehetetlenné teszi további szegmensek lefoglalását a halomterületből, ezáltal *fenntartja* a fennmaradó helyet a karakterlánc-terület számára.

Ha a fentivel ellentétes stratégiát alkalmazna, azaz folyamatosan csökkentené a szegmensméretet, amíg már egyetlen csomópontot sem tud lefoglalni (vagyis az "`(alloc 1) (expand 1)`" függvényhívás 0 értékkel tér vissza), akkor a *teljes* halom területet a csomópontok számára használná fel, ez azonban elégtelen karakterlánc-terület hibához vezetne. Ez nem ajánlott, mivel az AutoLISP elérhető karakterlánc-terület nélkül használhatalan lesz.

Virtuális függvénylapozás

A virtuális függvénylapozás csak akkor lép működésbe, ha már minden egyéb típusú virtuális memória kimerült, ami a legtöbb platformon csak ritkán következik be. A **vmon** függvény az AutoLISP előző verzióival való kompatibilitást biztosítja.

Ha az Ön AutoLISP alkalmazása túl nagyra nőtt, és már nem fér el a rendelkezésre álló csomóponti területen, akkor be kell kapcsolnia az AutoLISP *virtuális függvénylapozóját*, lehetővé téve ezzel a program további működését. Ehhez még a program első **defun** függvénydefiníciója előtt meg kell hívnia a **vmon** függvényt:

```
(vmon)
```

Ezzel lehetővé teszi a virtuális függvénylapozást az adott rajzszerkesztési alkalom hátralevő részében. Ha egyszer bekapcsolta a függvénylapozást, az már nem kapcsolható ki. A lapozásra csak a **defun** által, a **vmon** hívást követően definiált függvények alkalmasak, így amennyi-

ben a **vmon** hívása előtt definiált függvényeket, ezekre a függvénylapozás nem vonatkozik, és a továbbiakban is előidézhetik a program elakadását; ilyenkor az elégtelen node terület hibaüzenet jelenik meg.

Miután a **vmon** függvényt végrehajtotta, az AutoLISP a ritkán használt függvényeket kilapozza a memóriából, valahányszor kifogy a csomóponti területből; és automatikusan visszaolvassa őket, amikor újra szükség lesz rájuk. A függvények az AutoCAD fájllapozó által kezelt ideiglenes fájlokba kerülnek.

A virtuális lapozás csak a *függvényekre* vonatkozik; ettől függetlenül elegendő csomóponti terület kell, hogy maradjon a program által használt összes adatlista, a függvénynév és változónév számára.

A **mem** függvény két további mezőt is kijelez, ha a **vmon** be van kapcsolva. A *Beolvasott* mező azoknak a függvényeknek a számát tartalmazza, amelyeket a lapkezelő *beolvasott* a virtuális memória számára létrehozott lapfájlból. Ezek a visszakért függvények, amelyek meghívásukkor egyszerűen visszakerültek a lapfájlból. A *lapfájl* bejegyzés a kilapozott függvények tárolására létrehozott lapfájl méretét mutatja.

A **vmon** függvény végrehajtása után minden **defun** függvényhívás egy *lapozási táblának* nevezett új csomópontot illeszt a függvénylisták elejére, a formális argumentumok listája elé. A lapozási tábla csomópontokat kizárólag a lapkezelő használhatja, felhasználói programoknak semmiképpen sem szabad megváltoztatniuk. Ezen csomópontok esetében a **type** függvény PAGETB értékkel tér vissza.

Amikor az AutoLISP a rendelkezésre álló összes csomópontot felhasználta, a legritkábban használt függvényt kiírja a lapfájlba, a lapozási táblában megjegyezve a lapfájlban elfoglalt címét, és felszabadítja a függvény lapozási táblát követő összes csomópontját. A lapozási tábla bejelöli, hogy a függvényt kiírta a lapfájlba. Amikor egy kilapozott függvény kiértékelésére kerül sor, az AutoLISP a végrehajtást megelőzően visszaolvassa azt a lapfájlból (más függvényeket esetleg kilapoz). Amennyiben egy függvény egyszer már kikerült a lapfájlba, az ezt követő kilapozások egyszerűen csak felszabadítják a csomópontjait. A függvény kiírására ilyenkor nincs szükség, mivel már szerepel a fájlban.

Az AutoLISP **defun** függvényével létrehozott függvények olyan listák, amelyek bármely más listához hasonlóan feldolgozhatók. A feldolgozást végző programoknak figyelemmel kell kísérniük a lapkezelő működését (vagy ne használja a **vmon** függvényt). A **defun** függvénnyel létrehozott függvények elején egy lapozási tábla-csomópont található, amelyet a függvény vizsgálatakor Önnek át kell ugornia. Ha a függvényt listaként (a **defun** függvény megkerülésével) hozza létre, kifogástalanul fog működni, azonban kilapozásra nem lesz alkalmas, ezért könnyen kifogyhat a memóriából. Megfordítva, egy függvényt mindig a memóriában tarthat, ha újradefiniálja a lapozási táblája

nélkül. Például, a `zorp` nevű függvényt úgy rögzítheti a memóriában, ha a

```
(setq zorp (cdr zorp))
```

utasítással kitörli az élen álló lapozási tábláját. A függvény kinyomtatásakor a lapozási tábla szóközként jelenik meg. Azt, hogy egy függvény lapozható-e, vagy sem, úgy állapíthatja meg, hogy megnézi, szerepel-e szóköz az első baloldali zárójel után: ha igen, a függvény lapozható.

Amennyiben egy kilapozott függvényt adatként próbál végigpásztázni, a függvény listájában csak a lapozási táblát fogja látni. A függvény ugyanis ilyen hozzáférési kísérletek hatására nem kerül vissza a memóriába—csak *kiértékelés* esetén. Ha tehát új függvényt szerkeszt, és menet közben módosítani kívánja azt, akkor a **`defun`** használata helyett listaként állítsa össze, vagy alkalmazza a fenti módszert a memóriában való benntartásra.

16

ALKALMAZÁS-SPECIFIKUS DXF CSOPORTKÓDOK

Ez a fejezet azokat a DXF csoportkódokat tárgyalja, amelyek AutoLISP vagy ADS alkalmazásban fordulnak elő a leggyakrabban. A csoportkódok az első részben a kódszámok emelkedő sorrendjében, a második részben pedig objektumtípus szerinti bontásban szerepelnek. (További információ a csoportkódokat használó függvények leírásáról a 9., "Kiválasztási halmazokat, objektumokat és szimbólumtáblákat kezelő függvények" című fejezetben található.)

Az ebben a fejezetben tárgyalt csoportkódok hasonlóak a DXF fájlokban használtakhoz.

Csoportkódok emelkedő számsorrendben

A következő táblázatban negatív csoportkódok is szerepelnek, melyek a DXF fájlban nem, de az alkalmazásokban előfordulnak. Kimaradt viszont néhány olyan csoportkód, amely a programokban nem szerepel. A táblázatban a "(rögzített)" kifejezés azt jelenti, hogy az adott csoportkód mindig azonos szerepet tölt be. A nem rögzített csoportkódok szerepe változhat, aszerint, hogy milyen környezetben fordulnak elő.

A rajzelemek csoportkódja kódszám szerint

Csoportkód	*Leírás*
–5	Ismétlődő reaktor lánc
–4	Feltételes műveletjel (*csak* az `ssget` és `ads_ssget()` függvényekkel használható)
–3	Bővített rajzelemadat (XDATA) jelzője (rögzített)
–2	Rajzelemnév-hivatkozás (rögzített)
–1	Rajzelemnév. A rajz minden megnyitásakor változik. Soha nincs elmentve. (rögzített)
0	A rajzelem típusát jelző karakterlánc (rögzített)
1	A rajzelem elsődleges szöveges értéke
2	Egy név: attribútum cimkéje, blokk neve, stb.
3–4	Egyéb szöveges- vagy névértékek
5	Rajzelem-sorszám. Legfeljebb 16 hexadecimális számjegyből álló karakterlánc (rögzített)
6	Vonaltípus-név (rögzített)
7	Szövegstílus-név (rögzített)
8	Fólianév (rögzített)
10	Elsődleges pont (három valós számból álló lista). Vonal vagy szöveg rajzelem kezdőpontja, egy kör középpontja, stb. A DXF fájlokban ez a kód 10-es, 20-as és 30-as csoportkódként jelenik meg, melyek mindegyike valós szám.
11–18	Egyéb pontok. Az alkalmazások csak ezeket a csoportkódokat látják. A (20–28) és *Z* (30–38) koordináták, melyek a DXF fájlban szerepelnek, az alkalmazások számára egy AutoLISP pontlista részeként adódnak át.
39	A rajzelem vastagsága, amennyiben nullától különbözik (rögzített)

A rajzelemek csoportkódja kódszám szerint (*folytatás*)

Csoportkód	*Leírás*
40–48	Lebegőpontos értékek (szöveg magassága, méretarány-tényezők, stb)
49	Ismétlődő lebegőpontos érték. Egy változó hosszúságú tábla egyetlen elemében több 49-es csoport is előfordulhat (mint például a vonalszakasz hossza egy LTYPE táblában). Egy *7x* mindig megelőzi az első 49-es csoportot, megadva a tábla hosszát .
50–58	Szögek
62	Színszám (rögzített)
66	"Rajzelemek következnek" flag (rögzített)
67	Tér (modell- vagy papírtér) (rögzített)
70–78	Egész értékek, mint például ismétlések száma, flag-bitek vagy módok
90-99	32 bites egész értékek
100	Alosztály-adatok jelzője (a leszármazott osztály neve karakterláncként jelenik meg)
102	Tetszőleges karakterlánc, melyet "{*tetszőleges név*" vagy "}" követ
105	DIMVAR szimbólumtábla bejegyzésének objektumsorszáma
210	Kihúzási irány (rögzített). Akárcsak a pontok koordinátáinak esetében, az alkalmazások csak a 210-es csoportot látják. A kihúzási vektorok (220) és *Z* (230) összetevői az alkalmazások számára egy AutoLISP pontlista részeként adódnak át.
280-289	8 bites egész értékek
300-309	Tetszőleges karakterlánc
310-319	Tetszőleges bináris adag
320-329	Tetszőleges objektumsorszám
330-339	Gyenge pointer azonosító (mutatót ad meg, mely a rajzon más objektumokra mutat)
340-349	Erős pointer azonosító (mutatót ad meg, mely a rajzon más objektumokra mutat)
350-359	Gyenge tulajdonos azonosító (mutatót ad meg, mely a rajzon más objektumokra mutat)
360-369	Erős tulajdonos azonosító (mutatót ad meg, mely a rajzon más objektumokra mutat)
999	Megjegyzések

A rajzelemek csoportkódja kódszám szerint (*folytatás*)

Csoportkód	*Leírás*
1000	ASCII karakterlánc (legfeljebb 255 karakter hosszú) az XDATA-ban
1001	Bejegyzett alkalmazásnév (legfeljebb 31 karakter hosszú ASCII karakterlánc), amelyhez az XDATA tartozik (rögzített)
1002	XDATA vezérlő karakterlánc ("`{`" vagy "`}`") (rögzített)
1003	Fólianév az XDATA-ban
1004	Bájt adag (legfeljebb 127 bájt hosszú) az XDATA-ban
1005	Rajzelem-sorszám az XDATA-ban. Legfeljebb 16 hexadecimális számjegyből álló karakterlánc
1010	Egy pont az XDATA-ban (*X*, *Y* és *Z*)
1011	Világtérbeli háromdimenziós pozíció az XDATA-ban (*X*, *Y* és *Z*)
1012	Világtérbeli háromdimenziós elmozdulás az XDATA-ban
1013	Világtérbeli háromdimenziós irány az XDATA-ban. Az alkalmazások csak ezeket a koordináta csoportkódokat látják. Az *Y* (1020, 1021, 1022 vagy 1023) és *Z* (1030, 1031, 1032 vagy 1033) koordináták, melyek a DXF fájlban szerepelnek, az alkalmazások számára egy AutoLISP pontlista részeként adódnak át.
1040	Lebegőpontos érték az XDATA-ban
1041	Távolságérték az XDATA-ban
1042	Méretarány-tényező az XDATA-ban
1070	16 bites egész szám az XDATA-ban
1071	32 bites (long) előjeles egész szám az XDATA-ban

Grafikus rajzelemek

Az alábbi rész a grafikus rajzelemekre (objektumokra) vonatkozó csoportkódokat mutatja be. A nem-grafikus rajzelemek csoportkódjai (szimbólumtáblák és szótár bejegyzések) a "Nem grafikus rajzelemek" című résznél, a 590. oldalon találhatók.

Általános csoportkódok

Az alábbi táblázat azokat a csoportkódokat mutatja be, amelyek elméletileg az összes grafikus rajzelemre vonatkoznak. Az opcionális kódok szürke alapon jelennek meg. Ne felejtse el, hogy az itt felsorolt kódok olyan rajzelemtípus szerinti csoportkódok táblázatában is szerepelhetnek, amely a specifikus rajzelemekkel kapcsolatos kódokat tartalmazza.

Az összes grafikus rajzelemre vonatkozó csoportkódok

Csoportkód	*Jelentés*	*Elhagyásakor az alapértelmezés*
–1	Rajzelemnév. (A rajz minden megnyitásakor megváltozik.)	*Nem hagyható el*
0	Rajzelemtípus	*Nem hagyható el*
8	Fólianév	*Nem hagyható el*
5	Sorszám	*Nem hagyható el*
6	Vonaltípus-név (ha nem FÓLIA). A speciális BLOKK vonaltípus-név lebegő vonaltípust jelöl.	FÓLIA
48	Vonaltípus nagysága	1.0
62	Színszám (ha nem FÓLIA). A nulla a BLOKK (lebegő) színt jelöli. A 256 FÓLIA színt jelöli. A negatív érték azt jelenti, hogy a fólia ki van kapcsolva.	FÓLIA
67	Ha hiányzik vagy nulla, akkor a rajzelem a modelltérben van. Az 1 azt jelenti, hogy a rajzelem a papírtérben van.	0
90	ADE zár	

Megjegyzés *Ne* írjon olyan programot, amely erre a sorrendre támaszkodik. Bár ezek a táblázatok olyan sorrendben tartalmazzák a csoportkódokat, ahogyan azok általában következnek, ez azonban bizonyos feltételek mellett, illetve az AutoCAD későbbi verzióiban változhat. A rajzelemet vezérlő kódot esetválasztásos (switch) vagy táblázatos módon kell vezetni, hogy akkor is helyesen dolgozzon fel minden csoportot, ha azok nem a várt sorrendben követik egymást.

Rajzelemtípus szerinti csoportkódok

A következő táblázat egy DXF fájl ENTITIES részében található formában mutatja be a csoportkódokat. Az opcionális kódok szürke alapon szerepelnek. A blokkdefiníciókra, szimbólumtábla bejegyzésekre és szótár bejegyzésekre vonatkozó kódokról további információ található a "Nem-grafikus rajzelemek" című résznél, az 588. oldalon.

Rajzelemtípus szerinti csoportkódok

Rajzelemtípus	*Csoportkódok*	*Jelentés*
3DFACE	10	Első sarokpont
	11	Második sarokpont
	12	Harmadik sarokpont
	13	Negyedik sarokpont. (Amennyiben csak három sarokpont van megadva, ez megegyezik a harmadik sarokponttal.)
	70	Láthatalan él flagek (opcionális; alapértelmezése: 0): 1 Első él láthatalan. 2 Második él láthatalan. 4 Harmadik él láthatalan. 8 Negyedik él láthatalan.
3DSOLID	1	A fájlstruktúra első sora
	1	A fájlstruktúra további sorai
	3	A fájlstruktúra első sora (ha az előző 1-es csoport karakterlánca > 255 karakternél)
	3	További sorok (ha az előző 1-es csoport karakterlánca > 255 karakternél)
	1	A fájlstruktúra utolsó sora
ARC	10	Középpont
	40	Sugár
	50	Kiinduló szög
	51	Befejező szög
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
ATTDEF	10	Szöveg kezdőpontja
	40	Szöveg magassága
	1	Alapértelmezés szerinti érték (karakterlánc)
	3	Prompt karakterlánc
	2	Cím karakterlánc
	70	Attribútum flagek: 1 Láthatatlan attribútum (nem jelenik meg). 2 Konstans attribútum. 4 Az attribútum megadásakor ellenőrzés szükséges. 8 Előre beállított értékű attribútum (a beillesztés során nem jelenik meg prompt).
	73	Mező hossza (opcionális; alapértelmezése: 0)
	50	Szöveg elforgatási szöge (opcionális; alapértelmezése: 0)
	41	Relatív *X* irányú méretarány-tényező a Fit típusú szöveg számára (opcionális; alapértelmezése: 1)
	51	Dőlésszög (opcionális; alapértelmezése: 0)
	7	Szövegstílus-név (opcionális; alapértelmezése: STANDARD)
	71	Szöveggenerálási flagek(opcionális; alapértelmezése: 0)—lásd a TEXT rajzelemnél
	72	Vízszintes szövegigazítás típusa (opcionális; alapértelmezése: 0)—lásd a TEXT rajzelemnél
	74	Függőleges szövegigazítás típusa (opcionális; alapértelmezése: 0)—lásd a TEXT rajzelemnél
	11	Igazítási pont (opcionális; csak akkor szerepel, ha a 72-es vagy 74-es csoport jelen van, és nem nulla)
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
ATTRIB	10	Szöveg kezdőpontja
	40	Szöveg magassága
	1	Érték (karakterlánc)
	2	Cím karakterlánc
	70	Attribútum flagek: 1 Láthatatlan attribútum (nem jelenik meg). 2 Konstans attribútum. 4 Az attribútum megadásakor ellenőrzés szükséges. 8 Előre beállított értékű attribútum (a beillesztés során nem jelenik meg prompt).
	73	Mező hossza (opcionális; alapértelmezése: 0)
	50	Szöveg elforgatási szöge (opcionális; alapértelmezése: 0)
	41	Relatív *X* irányú méretarány-tényező a Fit típusú szöveg számára (opcionális; alapértelmezése: 1)
	51	Dőlésszög (opcionális; alapértelmezése: 0)
	7	Szövegstílus-név (opcionális; alapértelmezése: STANDARD)
	71	Szöveggenerálási flagek(opcionális; alapértelmezése: 0)—lásd a TEXT rajzelemnél
	72	Vízszintes szövegigazítás típusa (opcionális; alapértelmezése: 0)—lásd a TEXT rajzelemnél
	74	Függőleges szövegigazítás típusa (opcionális; alapértelmezése: 0)—lásd a TEXT rajzelemnél
	11	Igazítási pont (opcionális; csak akkor szerepel, ha a 72-es vagy 74-es csoport jelen van, és nem nulla)
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))
BODY	1	A fájlstruktúra első sora
	1	A fájlstruktúra további sorai
	3	A fájlstruktúra első sora (ha az előző 1-es csoport karakterlánca > 255 karakternél)

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
	3	További sorok (ha az előző 1-es csoport karakterlánca > 255 karakternél)
	1	A fájlstruktúra utolsó sora
CIRCLE	10	Középpont
	40	Sugár
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))
DIMENSION	2	A méret rajzi elemeit tartalmazó blokk neve
	3	Méretezési stílus neve
	10	Definíciós pont
	11	A méretszöveg középppontja
	12	Beillesztési pont a méretezés származékai (Bázisvonal és Folytat) számára
	70	A méretezés típusa. A 0 és 70 közötti kódok egész számok; nem összetettek. A 128-as kód a 0 és 70 közötti kódokkal kombinálódik, ha a méretszöveg nem az alapértelmezés szerinti helyen van 0 Forgat, vízszintes vagy függőleges 1 Illeszt 2 Szög 3 Átmérő 4 Sugár 5 Szög 3 pont 6 Y-típusú ordináta 70 *X*-típusú ordináta 128 Más kódokhoz adódik, ha a szöveg nem az alapértelmezés szerinti helyen van
	1	A felhasználó által megadott méretszöveg (opcionális; alapértelmezése: a számított érték)
	13	Definíciós pont a vonalas és szögméretezés esetében
	14	Definíciós pont a vonalas és szögméretezés esetében
	15	Definíciós pont átmérő-, sugár- és szögméretezés esetében

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
	16	Szögméretezés méretívének definíciós pontja
	40	Mutatóvonal hossza sugár- vagy átmérőméretezés esetében
	50	Elforgatott, vízszintes vagy függőleges vonalas méretezés szöge
	51	A vízszintes irány (opcionális)
	52	Segédvonal szöge dőlt vonalas méretezés esetében (opcionális)
	53	A méretszöveg elforgatási szöge (opcionális)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))
	–3	"ACAD" alkalmazás azonosító (ID). Ezzel kezdődik az xdata azon szakasza, amely leír bármilyen, a rajzelemre vonatkozó méretezési felülírást. Lásd "A méretezési stílus felülírása" című résznél, a 588. oldalon.
ELLIPSE	10	Középpont
	11	A nagy tengely végpontja (a középponthoz viszonyítva)
	40	A kis tengely hossza (a nagy tengely hosszának százaléka)
	41	Induló paraméter
	42	Befejező paraméter
INSERT	66	Attribútumok következnek flag változó (opcionális; alapértelmezése: 0)
	2	Blokknév
	10	Beillesztési pont
	41	*X* méretarány-tényező (opcionális; alapértelmezése: 1)
	42	*Y* méretarány-tényező (opcionális; alapértelmezése: 1)
	43	*Z* méretarány-tényező (opcionális; alapértelmezése: 1)
	50	Elforgatási szög (opcionális; alapértelmezése: 0)
	70	Oszlopok száma (opcionális; alapértelmezése: 1)
	71	Sorok száma (opcionális; alapértelmezése: 1)
	44	Oszlopok távolsága (opcionális; alapértelmezése: 0)

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
	45	Sorok távolsága (opcionális; alapértelmezése: 0)
	–3	"ACAD" alkalmazás azonosító (ID). Ezzel kezdődik az xdata azon része, amely a sraffozás tulajdonságait tartalmazza.
LEADER	2	Méretezési stílus neve
	72	Segédvonal típusa 0 Egyenes vonalakból álló szakaszok 1 Spline
	73	A csomópontok száma az útvonalon
	10	Csomópont koordináták (több bejegyzés)
	11	*X* tengely irányvektora (VKR)
	12	*Z* tengely irányvektora (VKR)
	340	Erős hivatkozás a csatolt szöveg rajzelemre
	–3	"ACAD" alkalmazás azonosító (ID). Ezzel kezdődik az xdata azon szakasza, amely leír bármilyen, a rajzelemre vonatkozó méretezési felülírást. Lásd "A méretezési stílus felülírása" című résznél, az 588. oldalon.
LINE	10	Kezdőpont
	11	Végpont
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))
MLINE	7	Stílusnév (maximum 32 karakterből álló karakterlánc. Ez az mline rajzelemhez használt stílus neve. A stílus bejegyzésének szerepelnie kell az ACAD_MLINESTYLE szótárban.)
	41	Méretarány-tényező (valós)
	73	Csomópontok száma (egész)
	10	Csomópont koordináták (több bejegyzés—egy bejegyzés egy csomópont számára). Minden csomópont szegmenst egy mline szegmens paraméterezési adata követ.

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
MTEXT	1	Karakterlánc. Ha a karakterlánc rövidebb 255 karakternél, az összes karakter az 1-es csoportban jelenik meg. Ha a karakterlánc hosszabb mint 255 karakter, akkor azt felosztja 255 karakterből álló adagokra, amelyek egy vagy több 3-as csoportkódban helyezkednek el. Ha használja a 3-as csoportkódot, akkor a maradék szöveget az 1-es csoportkódba helyezi el.
	3	További szöveg (mindig 255 karakteres adagokban, opcionális)
	7	Szövegstílus-név (STANDARD ha nincs megadva)
	10	Beillesztési pont
	11	*X* tengely irányvektora (VKR)
	12	*Z* tengely irányvektora (VKR)
	41	Hivatkozási négyszög szélessége
	71	Illesztési pont: 1 BalFelső 2 KözépsőFelső 3 JobbKözépső 4 BalFelező 5 KözépsőFelező 6 JobbFelező 7 BalAlsó 8 AlsóKözépső 9 JobbAlsó
	72	Rajzolási irány: 1 Balról jobbra 2 Jobbról balra 3 Felülről lefele 4 Alulról felfele
OLEFRAME	(nincs csoportja)	Az oleframe rajzelemeknek nincs speciális DXF csoportkódjuk
POINT	10	Pont
	50	A pont megrajzolásakor aktuális FKR *X* tengelyének szöge (opcionális; alapértelmezése: 0). Abban az esetben használatos, ha a PDMODE értéke nem nulla.
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
POLYLINE	66	Csomópontok következnek flag (mindig 1 a vonalláncok esetében)
	10	Egy fiktív pont. Az *X* és *Y* koordinátája mindig 0, a *Z* koordináta pedig a vonallánc kiemelési szintjét határozza meg.
	70	Vonallánc flag (bit-kódolású, tehát kombinálható) (opcionális; alapértelmezése: 0): 1 Zárt vonallánc (vagy *M* irányban zárt síkháló felület). 2 Görbeillesztő csomópontok hozzáadva. 4 Spline-illesztő csomópontok hozzáadva. 8 Háromdimenziós vonallánc. 16 Háromdimenziós síkháló felület. 32 A síkháló felület *N* irányban zárt. 64 Ez a vonallánc egy soklapháló. 128 A vonaltípus mintázatának kirajzolása folyamatos ezen vonallánc csomópontjainak környezetében.
	40	Alapértelmezés szerinti kezdő szélesség (opcionális; alapértelmezése: 0)
	41	Alapértelmezés szerinti befejező szélesség (opcionális; alapértelmezése: 0)
	71	A síkháló felület *M* csomópontjainak száma (opcionális; alapértelmezése: 0)
	72	A síkháló felület *N* csomópontjainak száma (opcionális; alapértelmezése: 0)
	73	Simuló felület *M* sűrűsége (opcionális; alapértelmezése: 0)
	74	Simuló felület *N* sűrűsége (opcionális; alapértelmezése: 0)
	75	Görbék és simuló felületek típusa (opcionális; alapértelmezése: 0)—ezek egész típusú kódok; nem bit-kódolásúak: 0 Nincs illesztve simuló felület 5 Másodfokú B-spline felület 6 Harmadfokú B-spline felület 8 Bezier felület
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
RAY	10	Bázispont
	11	Irányvektor
REGION	1	A fájlstruktúra első sora
	1	A fájlstruktúra további sorai
	3	A fájlstruktúra első sora (ha az előző 1-es csoport karakterlánca > 255 karakternél)
	3	További sorok (ha az előző 1-es csoport karakterlánca > 255 karakternél)
	1	A fájlstruktúra utolsó sora
SEQEND	–2	A sorozat kezdő elemének a neve
SHAPE	10	Beillesztési pont
	40	Méret
	2	Alaknév
	50	Elforgatási szög (opcionális; alapértelmezése: 0)
	41	Relatív *X* irányú méretarány-tényező (opcionális; alapértelmezése: 1)
	51	Dőlésszög (opcionális; alapértelmezése: 0)
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))
SOLID	10	Első sarokpont
	11	Második sarokpont
	12	Harmadik sarokpont
	13	Negyedik sarokpont. (Amennyiben csak három sarokpont van megadva, ez megegyezik a harmadik sarokponttal.)
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
SPLINE	70	A spline görbe foka
	71	Spline flag (bit értékek): 1 Zárt spline 2 Periodikus spline 4 Racionális spline 8 Sík 16 Lineáris (A sík bit is be van állítva.)
	72	Csomók száma
	73	Vezérlőpontok száma
	74	Illesztő pontok száma (ha léteznek)
	42	Illesztés tűrése
	43	Vezérlőpont tűrése
	44	Illesztés tűrése (amennyiben illesztő spline)
	12	Induló érintő (opcionális)
	13	Befejező érintő (opcionális)
	40	Csomó érték(ek) (Több 40-es csoport bejegyzés létezhet.)
	10	Vezérlőpont(ok) (Több 10-es csoport bejegyzés létezhet.)
	11	Illesztő pont(ok) (Több 11-es csoport bejegyzés létezhet.)
TEXT	10	Beillesztési pont
	40	Magasság
	1	Szövegérték (maga a karakterlánc)
	50	Elforgatási szög (opcionális; alapértelmezése: 0)
	41	Relatív *X* irányú méretarány-tényező. Fit típusú szöveg számára (opcionális; alapértelmezése: 1.0)
	51	Dőlésszög (opcionális; alapértelmezése: 0)
	7	Szövegstílus-név (opcionális; alapértelmezése: STANDARD)
	71	Szöveggenerálási flagek (opcionális; alapértelmezése: 0): 2 Szöveg visszafelé (*X* irányban tükrözve) 4 Szöveg fejjel lefelé (*Y* irányban tükrözve)

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
	72	Vízszintes szövegigazítás (opcionális; alapértelmezése: 0)—egész típusú kódok; *nem* bit-kódolásúak: 0 Bal 1 Közép 2 Jobb 3 Igazít (ha a függőleges igazítás = 0) 4 Közép (ha a függőleges igazítás = 0) 5 Illeszt (ha a függőleges igazítás = 0)
	73	Függőleges szövegigazítás (opcionális; alapértelmezése: 0)—egész típusú kódok; *nem* bit-kódolásúak: 0 Alapvonal 1 Közép 2 Felező 3 Felső
	11	Igazítási pont (opcionális; csak akkor szerepel, ha a 72-es vagy 73-as csoport jelen van és nem nulla)
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))
TOLERANCE	10	Beillesztési pont
	11	*X* tengely irányvektora (VKR)
	12	*Z* tengely irányvektora (VKR)
	1	Elsődleges szövegérték
	3	Méretezési stílusnév
	–3	"ACAD" alkalmazás azonosító (ID) . Ezzel kezdődik a bővített adatok (xdata) azon szakasza, amely leír bármilyen, a rajzelemre vonatkozó méretezési felülírást. Lásd "A méretezési stílus felülírása" című résznél, az 588. oldalon.
TRACE	10	Első sarokpont
	11	Második sarokpont
	12	Harmadik sarokpont
	13	Negyedik sarokpont

Rajzelemtípus szerinti csoportkódok (*folytatás*)

Rajzelemtípus	*Csoportkódok*	*Jelentés*
	39	Vastagság (opcionális; alapértelmezése: 0)
	210	Kihúzási irány. Akkor van jelen, ha a rajzelem kihúzási iránya nem párhuzamos a Világ koordináta-rendszer Z tengelyével. (opcionális; alapértelmezése (0,0,1))
VERTEX	10	Hely
	40	Kezdő szélesség (opcionális; alapértelmezése: 0)
	41	Befejező szélesség (opcionális; alapértelmezése: 0)
	42	Kihajlás (opcionális; alapértelmezése: 0)
	70	Csomópont flagek (opcionális; alapértelmezése: 0): 1 Görbeillesztés által létrehozott extra csomópont 2 Erre a csomópontra definiált görbeillesztési érintő. A 0 értékű görbeillesztési érintő iránya kimaradhat a DXF kimenetből, lényeges viszont akkor, ha ez a bit be van állítva 4 *Nem használt* 8 Spline-illesztés által létrejött spline-csomópont 16 Spline-keret vezérlőpont 32 Háromdimenziós vonallánc csomópontja 64 Háromdimenziós síkháló-felület csomópontja 128 Soklapháló-felület csomópontja
	50	Görbeillesztési érintő iránya (opcionális)
VIEWPORT	10	Középpont
	40	Szélesség papírtéri egységekben
	41	Magasság papírtéri egységekben
	69	Nézetablak azonosító (ID). (A rajz minden megnyitásakor változik, soha nincs elmentve --a papírtérbeli nézetablakot kivéve, amelynek értéke mindig 1.)
	68	Nézetablak-állapot mező
	–3	"ACAD" alkalmazás azonosító (ID). A nézetablakot leíró XDATA szakasz elején áll.
XLINE	10	Bázispont
	11	Irányvektor

A méretezési stílus felülírása

Méretezési felülírások méretezés, segédvonal és tűrés rajzelemek esetében alkalmazhatók. Ezen rajzelemekre vonatkozó bármilyen felülírás bővített adatként (xdataként) tárolódik a rajzelemben. A felülírt méretezési változók csoportkódjai és a hozzájuk tartozó értékek az 1002-es csoporotkódban párosan szerepelnek. A következő példa egy méretezési rajzelem xdata elemét mutatja be, ahol a DIMTOL és DIMCLRE változók kerültek felülírásra.

```
(setq diment (car (entsel)))          Kiválasztja a méretezési rajzelemet
(setq elst (entget diment '("ACAD"))) Lekérdezi a rajzelem adatlistáját
(assoc -3 elst)                       Csak az xdata részt vonja ki
```

A fenti programrészlet a következőket adja vissza:

```
(-3 ("ACAD" (1000 . "DSTYLE")
(1002 . "{")(1070 . 177)(1070 . 3)(1002 . "}")
(1002 . "{")(1070 . 71)(1070 . 1)(1002 . "}")
))
```

Nem-grafikus rajzelemek

Ez a rész a nem-grafikus rajzelemekre vonatkozó csoportkódokat mutatja be. A grafikus rajzelemek (objektumok) csoportkódjait a "Grafikus rajzelemek" című résznél, a 575. oldalon tárgyalja.

Általános csoportkódok

Az alábbi táblázatban bemutatott csoportkódok tulajdonképpen az összes nem-grafikus rajzelemre vonatkoznak. Az opcionális kódok szürke alapon jelennek meg. Ne felejtse el, hogy az alább felsorolt kódok olyan rajzelemtípus szerinti csoportkódok táblázatában is szerepelhetnek, amely a specifikus rajzelemekkel kapcsolatos kódokat tartalmazza.

Az összes nem-grafikus rajzelemre vonatkozó csoportkódok

Csoport-kód	*Jelentés*	*Elhagyásakor az alapértelmezés*
–1	Rajzelenév. (A rajz minden megnyitásakor változik.)	*Nem* hagyható el
0	Rajzelem típusa	*Nem* hagyható el
5	Sorszám	*Nem* hagyható el

Az összes nem-grafikus rajzelemre vonatkozó csoportkódok (*folytatás*)

Csoport-kód	*Jelentés*	*Elhagyásakor az alapértelmezés*
70	Flag értékek 16 a táblabejegyzés külső függőségben van egy xreffel 32 a külsőleg függő xref sikeresen feloldott 64 a rajz utolsó szerkesztése alkalmával a tábla bejegyzésre legalább egy rajzelem hivatkozott. (Ez a flag az AutoCAD parancsok számára van jelen; a DXF fájlt beolvasó legtöbb program figyelmen kívül hagyhatja, és a DXF fájlt kiíró programoknak sem kell beállítaniuk ezt.)	
90	ADE zár	

Megjegyzés *Ne* írjon olyan programot, amely erre a sorrendre támaszkodik. Bár ezek a táblázatok olyan sorrendben tartalmazzák a csoportkódokat, ahogyan azok általában következnek, ez azonban bizonyos feltételek mellett, illetve az AutoCAD későbbi verzióiban változhat. A rajzelemet vezérlő kódot esetválasztásos (switch) vagy táblázatos módon kell vezetni, hogy akkor is helyesen dolgozzon fel minden csoportot, ha azok nem a várt sorrendben követik egymást.

Szótár csoportkódok

Az alábbi táblázat a szótár bejegyzések csoportkódjait tartalmazza. Ezek megegyeznek a DXF fájlok OBJECTS részében találhatóakkal. Az opcionális kódok szürke alapon jelennek meg.

Szótár csoportkódok

Rajzelem típusa	*Csoportkód*	*Jelentés*
DICTIONARY	90	Az ebben az objektumrekordban megjelenő bejegyzések maximális száma
	300	Rajzelem neve (minden bejegyzésnél ismétlődik)
	360	Az objektum bejegyzés sorszáma (minden bejegyzésnél ismétlődik)
ACAD_GROUP	2	Csoportnév
	3	Csoportleírás
	70	Választhatósági flag
	240	A csoportban található rajzelemek indexeinek száma
	5	A csoportban található rajzelemek sorszámai (több bejegyzés; a csoportban található minden rajzelemhez egy)

Szótár csoportkódok (*folytatás*)

Rajzelem típusa	*Csoportkód*	*Jelentés*
ACAD_ MLINESTYLE	3	Stílusleírás (256 karakternél rövidebb karakterlánc)
	51	Kiinduló szög (valós, radián, alapértelmezése: 90 fok.)
	52	Befejező szög (valós, radián, alapértelmezése: 90 fok.)
	70	flagek (bit-kódolású) 1= kitöltés 2 = "miterek" megjelenítése
	62	Szín kitöltés
	71	Kezdeti lezáró elem attribútum (bit-kódolású) 1 = vonalat rajzol 2 = belső íveket rajzol 4 = külső ívet rajzol
	72	Befejező befejező lezáró elem attribútum (bit-kódolású) 1 = vonalat rajzol 2 = belső íveket rajzol 4 = külső ívet rajzol
	73	Az ebben a stílusban szereplő elemek száma (egész típusú)
	49	Elemeltolás (valós, nincs alapértelmezése). Több bejegyzés létezhet—minden elem számára egy.
	62	Elemszín (egész, alapértelmezése: 0). Több bejegyzés létezhet—minden elem számára egy.
	6	Elemvonaltípus (karakterlánc, alapértelmezése: CONTINUOUS). Több bejegyzés létezhet—minden elem számára egy.

Blokk és szimbólumtábla csoportkódok

A következő táblázat a blokkdefiníciók és szimbólumtábla-bejegyzések csoportkódjait tartalmazza. Ezek megegyeznek a DXF fájlok TABLES és BLOCKS részében találhatóakkal. Az opcionális kódok szürke alapon jelennek meg.

Blokk és tábla csoportkódok

Rajzelem típusa	*Csoportkódok*	*Jelentés*
APPID	2	Felhasználó által bejegyzett alkalmazásnév (az XDATA számára)
	70	Szabvány flag értékek

Blokk és tábla csoportkódok (*folytatás*)

Rajzelem típusa	*Csoportkódok*	*Jelentés*
BLOCK	A blokkleírások a grafikus rajzelemekkel közös elemcsoportokat is tartalmazzák	
	2	Blokknév
	70	Típus flag (bit-kódolású értékek, kombinálhatók): 1 Ez egy névtelen blokk (melyet sraffozás, asszociatív méretezés, más belső művelet vagy egy alkalmazás hozott létre) 2 Változó attribútumok következnek 4 Ez a blokk egy külső referencia (Xref) 8 *Nem használt* 16 Ez a blokk külső függőségben áll 32 Ez egy feloldott külső referencia, vagy függőségben áll egy külső referenciával 64 Hivatkozás van erre a definícióra
	10	Bázispont
	1	Xref útvonalnév (opcionális; csak akkor szerepel, ha a blokk egy xref)
DIMSTYLE	2	Méretezési stílus neve
	70	Szabvány flag értékek
	170	DIMALT
	171	DIMALTD
	143	DIMALTF
	274	DIMALTTD
	286	DIMALTTZ
	273	DIMALTU
	285	DIMALTZ
	4	DIMAPOST
	41	DIMASZ
	275	DIMAUNIT
	5	DIMBLK
	6	DIMBLK1

Blokk és tábla csoportkódok (*folytatás*)

Rajzelem típusa	*Csoportkódok*	*Jelentés*
	7	DIMBLK2
	141	DIMCEN
	176	DIMCLRD
	177	DIMCLRE
	178	DIMCLRT
	271	DIMDEC
	46	DIMDLE
	43	DIMDLI
	44	DIMEXE
	42	DIMEXO
	287	DIMFIT
	147	DIMGAP
	280	DIMJUST
	144	DIMLFAC
	72	DIMLIM
	3	DIMPOST
	45	DIMRND
	173	DIMSAH
	40	DIMSCALE
	281	DIMSD1
	282	DIMSD2
	75	DIMSE1
	76	DIMSE2
	175	DIMSOXD
	77	DIMTAD
	272	DIMTDEC

Blokk és tábla csoportkódok (*folytatás*)

Rajzelem típusa	*Csoportkódok*	*Jelentés*
	146	DIMTFAC
	73	DIMTIH
	174	DIMTIX
	48	DIMTM
	172	DIMTOFL
	74	DIMTOH
	71	DIMTOL
	283	DIMTOLJ
	47	DIMTP
	142	DIMTSZ
	145	DIMTVP
	340	DIMTXSTY
	140	DIMTXT
	284	DIMTZIN
	270	DIMUNIT
	288	DIMUPT
	78	DIMZIN
ENDBLK	(Nincs csoportja)	Blokkdefiníció vége (*csak* a BLOCKS táblában fordul elő)
LAYER	2	Fólianév
	70	Fólia flagek: 1 Ha be van állítva, a fólia fagyasztott, egyébként a fólia ki van olvasztva 2 Ha be van állítva, a fólia az új nézetablakokban alapértelmezés szerint fagyasztott lesz 4 Ha be van állítva, a fólia lezárt
	62	Színszám (ha negatív, a fólia ki van kapcsolva)
	6	Vonaltípus

Blokk és tábla csoportkódok (*folytatás*)

Rajzelem típusa	*Csoportkódok*	*Jelentés*
LTYPE	2	Vonaltípus-név
	70	Szabvány flag értékek
	3	A vonaltípus leíró szövege
	72	Igazítás kódja
	73	Szakaszhossz tételek száma
	40	A minta teljes hossza
	49	Szakaszhossz (opcionális; ismétlődhet)
STYLE	2	Stílusnév
	70	Szabvány flag értékek
	40	Rögzített szövegmagasság
	41	Szélességi tényező
	50	Dőlésszög
	71	Szöveggenerálási flagek (bit-kódolású értékek, kombinálhatók): 2 Szöveg visszafelé (*X* irányban tükrözve) 4 Szöveg fejjel lefelé (*Y* irányban tükrözve)
	42	Utoljára használt magasság
	3	Elsődleges fontfájl neve
	4	Big-fontfájl neve (ha nincs, üres karakterlánc)
UCS	2	FKR név
	70	Szabvány flag értékek
	10	Origó (a VKR-ben)
	11	*X* tengely iránya (a VKR-ben)
	12	*Y* tengely iránya (a VKR-ben)
VIEW	2	Nézet neve
	70	Nézet flag: 1 Ha be van állítva, ez a papírtérbeli nézet
	40	Magasság

Blokk és tábla csoportkódok (*folytatás*)

Rajzelem típusa	*Csoportkódok*	*Jelentés*
	41	Szélesség
	10	Középpont (egy kétdimenziós pont) (a KKR-ben)
	11	Nézőirány a célpontból, VKR-ben
	12	Célpont, VKR-ben
	42	Fókusztávolság
	43	Elülső vágó sík
	44	Hátsó vágó sík
	50	Elforgatási szög
	71	Nézet mód—megegyeznek a VIEWMODE rendszerváltozó értékeivel
VPORT	2	Nézetablak neve (*Nem szükségszerűen egyedi.* Az aktuális nézet-ablak-konfigurációban minden nézetablak neve *ACTIVE, és a tábla első *ACTIVE nézetablaka az, amelyik éppen látható.)
	70	Szabvány flag értékek
	10	Bal alsó sarok (kétdimenziós pont)
	11	Jobb felső sarok (kétdimenziós pont)
	12	Középpont (kétdimenziós pont)
	13	A raszter alappontja (kétdimenziós pont)
	14	A raszter osztása (*X* és *Y*)
	15	A háló osztása (*X* és *Y*)
	16	Irány a célponttól
	17	Célpont
	40	Magasság
	41	Az oldalak aránya
	42	Fókusztávolság
	43	Elülső vágó sík
	44	Hátsó vágó sík

Blokk és tábla csoportkódok (*folytatás*)

Rajzelem típusa	*Csoportkódok*	*Jelentés*
	50	A raszter elforgatási szöge
	51	Elforgatási szög
	68	Állapot mező
	69	ID (azonosító)
	71	Nézet mód—megegyeznek a VIEWMODE rendszerváltozó értékeivel
	72	Kör zoom százalékban
	73	Gyors zoom beállítása
	74	UCSICON beállítása
	75	Raszter be/ki
	76	Háló be/ki
	77	Raszter stílusa
	78	Raszter izopár

17

AUTOLISP HIBAKÓDOK ÉS HIBAÜZENETEK

Ez a fejezet az AutoLISP hibakódok és hibaüzenetek leírását tartalmazza.

Hibakódok

A következő táblázat az AutoLISP által generált hibakódok értékeit mutatja be. Az ERRNO rendszerváltozó ezen értékek egyikét veszi fel, ha egy függvényhívás az AutoCAD által felismert hibát eredményez. Az AutoLISP alkalmazások az ERRNO rendszerváltozó aktuális értékét a `(getvar "errno")` függvényhívással tudják megvizsgálni.

Az ERRNO rendszerváltozó értéke nincs minden esetben 0-ra állítva, ezért közvetlenül azután kell megvizsgálni, miután egy AutoLISP függvény hibát jelentett; ellenkező esetben az általa jelzett hiba félrevezető lehet. Egy rajz indításakor vagy megnyitásakor a változó értéke mindig nulla.

Megjegyzés Az ERRNO lehetséges értékei és jelentésük a későbbi AutoCAD verziókban megváltozhatnak.

Programfutás közben fellépő hibák kódjai

Érték	*Jelentés*
1	Érvénytelen szimbólumtábla-név
2	Érvénytelen elemnév vagy kiválasztási halmaznév
3	A kiválasztási halmazok maximális számának túllépése
4	Érvénytelen kiválasztási halmaz
5	Blokkdefiníció helytelen használata
6	Külső referencia (Xref) helytelen használata
7	Objektumkiválasztás: a kijelölés sikertelen
8	Rajzelemfájl vége
9	Blokkdefiníciós fájl vége
10	Nem sikerült megtalálni az utolsó rajzelemet
11	Jogosulatlan kísérlet a nézetablak-objektum kitörlésére
12	A művelet VLÁNC közben nem megengedett
13	Érvénytelen sorszám
14	A sorszámozás nincs bekapcsolva
15	Érvénytelen argumentumok a koordináta-rendszerek transzformálási parancsában
16	Érvénytelen terület a koordináta-rendszer transzformálási parancsában

Programfutás közben fellépő hibák kódjai (*folytatás*)

Érték	*Jelentés*
17	Törölt rajzelem érvénytelen használata
18	Érvénytelen táblanév
19	Érvénytelen táblafüggvény-argumentum
20	Kísérlet csak olvasható változó beállítására
21	A nulla érték nem megengedett
22	Az érték a megengedett tartományon kívül esik
23	Összetett REGEN (regenerálás) van folyamatban
24	Kísérlet rajzelem típusának megváltoztatására
25	Rossz fólianév
26	Rossz vonaltípus-név
27	Rossz színnév
28	Rossz szövegstílus-név
29	Rossz alaknév
30	A rajzelemnek nem megfelelő mező
31	Kísérlet törölt rajzelem módosítására
32	Kísérlet seqend alárendelt rajzelem módosítására
33	Kísérlet sorszám megváltoztatására
34	Kísérlet nézetablak láthatóságának módosítására
35	Lezárt fólián lévő rajzelem
36	Rossz elemtípus
37	Rossz vonallánc-rajzelem
38	Befejezetlen összetett rajzelem a blokkban
39	Érvénytelen blokknév-mező
40	Megkettőzött blokk flag mező
41	Megkettőzött blokknév-mező
42	Rossz normálvektor

Programfutás közben fellépő hibák kódjai (*folytatás*)

Érték	*Jelentés*
43	Hiányzó blokknév
44	Hiányzó blokk flagek
45	Érvénytelen névtelen blokk
46	Érvénytelen blokkdefiníció
47	Hiányzó kötelező mező
48	Ismeretlen bővített adat (XDATA) -típus
49	Helytelen listabeágyazás az XDATA-ban
50	Hibásan elhelyezett APPID mező
51	A maximális XDATA méret túllépése
52	Elemkiválasztás: üres válasz
53	Megkettőzött APPID
54	Kísérlet a nézetablak-rajzelem létrehozására vagy módosítására
55	Kísérlet xref, xdef vagy xdep létrehozására vagy módosítására
56	`ssget` szűrő: a lista nem várt módon ért véget
57	`ssget` szűrő: a vizsgálat operandusa hiányzik
58	`ssget` szűrő: műveleti jel (–4) karakterlánca érvénytelen
59	`ssget` szűrő: helytelen beágyazás vagy üres feltétel
60	`ssget` szűrő: a feltétel eleje és vége nem egyezik
61	`ssget` szűrő: helytelen számú argumentum a feltételben (NOT vagy XOR esetében)
62	`ssget` szűrő: a beágyazhatóság felső határának túllépése
63	`ssget` szűrő: érvénytelen csoportkód
64	`ssget` szűrő: érvénytelen karakterlánc-vizsgálat
65	`ssget` szűrő: érvénytelen vektorvizsgálat
66	`ssget` szűrő: érvénytelen valósszám-vizsgálat
67	`ssget` szűrő: érvénytelen egészszám-vizsgálat

Programfutás közben fellépő hibák kódjai (*folytatás*)

Érték	*Jelentés*
68	A digitalizáló eszköz nem tábla
69	A tábla nincs kalibrálva
70	Érvénytelen argumentumok
71	ADS hiba: új eredménypuffer allokációja nem lehetséges
72	ADS hiba: null pointer észlelése
73	A futtatható fájl nem nyitható meg
74	Az alkalmazás már be van töltve
75	Maximális számú alkalmazás már betöltve
76	Az alkalmazás végrehajtása nem lehetséges
77	Nem kompatibilis verziószám
78	A beágyazott alkalmazás nem törölhető a memóriából
79	Az alkalmazás nem törlődik a memóriából
80	Az alkalmazás jelenleg nincs betöltve
81	Nincs elegendő memória az alkalmazás betöltéséhez
82	ADS hiba: érvénytelen transzformációs mátrix
83	ADS hiba: érvénytelen szimbólumnév
84	ADS hiba: érvénytelen szimbólumérték
85	Tiltott AutoLISP/ADS művelet kísérlete egy párbeszédpanel megjelenített állapotában

Hibaüzenetek

Amennyiben az AutoLISP hibát észlel, megszakítja a folyamatban lévő függvényt, és a hiba típusára utaló üzenettel meghívja a felhasználói `*error*` függvényt. Ha nincs definiálva felhasználói `*error*` függvény (vagy az `*error*` értéke nulla), akkor az üzenetet a szabvány hibaművelet az alábbi formában jeleníti meg:

hiba: *üzenet*

Az üzenetet a függvény fokozatos visszafejtése követi. Ha létezik felhasználói `*error*` függvény, a visszafejtés nem jön létre. Helyette a felhasználói függvényt hívja meg egyetlen *üzenet* argumentummal.

Felhasználói programok hibái

Az alábbiakban azon hibaüzeneteket soroljuk fel, amelyekkel időről időre találkozni fog AutoLISP függvények írása és javítása során. A legtöbb ilyen üzenet az alábbi jellegzetes AutoLISP programozási hibákra utal:

- Hibásan írt függvények vagy szimbólumnevek
- Rossz típusú vagy számú argumentum megadása
- Páratlan számú zárójel
- Páratlan számú idézőjel (befejezetlen karakterlánc)
- Kísérlet egy függvény eredményének felhasználására, még mielőtt sikerült volna ellenőrizni a függvény tökéletes lefutását

Jóllehet ezek az üzenetek általában a felhasználó programozási hibáit jelzik, magának az AutoLISP nyelvnek a programhibáiból is származhatnak. Ha nem talál semmilyen hibát a programjában, kérjük töltsön ki egy Hibajelentést (Bug Report), és küldje el az Autodesk részére a programmal együtt (lehetőleg floppy lemezen) .

az argumentumok száma nem megfelelő

A **quote** függvénynek kizárólag egy argumentuma lehet, mégis többel látta el.

AutoLISP verem túlcsordulás

Az AutoLISP veremtárolója megtelt. Ezt okozhatja túl sok rekurzív függvényhívás, vagy nagyon hosszú argumentumlista.

az AutoCAD visszautasította a függvényt

Valamely AutoCAD függvénynek átadott argumentumok érvénytelenek (például, ha a **setvar** függvénnyel egy csak olvasható rendszerváltozót próbál beállítani, vagy a **tblnext** függvényt érvénytelen táblanévvel használja), esetleg maga a függvény érvénytelen az aktuális összefüggésében. Például a **getxxx** felhasználói bevitelre szolgáló függvényeket nem használhatja a **command** függvényen belül.

bázispont szükséges

A **getcorner** függvényt a szükséges bázispont-argumentum nélkül hívta meg.

Boole arg1 0 vagy 15

A Boole függvény első argumentumának 0 és 15 közé eső egésznek kell lennie.

defun argumentumai nem lehetnek azonos nevűek

Azonos nevű argumentumokkal definiált függvény hibát okoz, és a fenti üzenetet generálja.

elégtelen karakterlánc-terület

A halomterület nem elegendő az adott karakterlánc befogadásához. További információ a "Memóriakezelés" című, 15. fejezetben található.

elégtelen node terület

A halomterület nem elegendő a kívánt művelet befogadásához. További információ a "Memóriakezelés" című, 15. fejezetben található.

érvénytelen argumentum

Az argumentum típusa nem megfelelő, vagy az értelmezési tartományon kívül esik.

érvénytelen argumentumlista

Hibás argumentumlistát adott át a függvénynek.

érvénytelen egész érték

Az AutoLISP olyan számot talált, amelynek értéke kisebb, mint a legkisebb egész, vagy nagyobb, mint a legnagyobb egész.

érvénytelen kapcsolt pár

A kapcsolt párok két elemből álló listák, és ezt a két elemet egy *szóköz-pont-szóköz* szerkezet választja el. A hibát az is okozhatja, ha egy valós szám tizedesponttal kezdődik; ilyen esetekben mindig írja ki a kezdő nullát is.

érvénytelen karakter

Egy kifejezés oda nem illő karaktert tartalmaz.

fájl nincs megnyitva

Az I/O művelet fájlleírója nem egy nyitott fájlhoz tartozik.

fájlméret túllépése

A fájl mérete túllépte az operációs rendszer által megengedett értéket.

fájlolvasás—elégtelen karakterlánc-terület

A fájl olvasása közben a karakterlánc-terület kimerült. További információ a "Memóriakezelés" című, 15. fejezetben található.

felesleges jobb zárójel

Egy vagy több felesleges zárójel.

függvény megszakítva

A felhasználó egy adatbeviteli promptra Ctrl+C vagy Esc (megszakít) billentyűkkel válaszolt.

a függvény argumentumra nem definiált

A `log` vagy `sqrt` függvény argumentuma az értelmezési tartományon kívül esik.

a függvénynek az argumentumok száma nem megfelelő

A felhasználói függvénynek átadott argumentumok száma nem egyezik a defun-ban meghatározottakkal.

a függvény valós számra nem definiált

Egy egész számot váró függvény valós számot kapott argumentumként—például `(1sh val 1.2)`.

grvecs végpont hiányzik

A `grvecs` függvénynek átadott vektorlistában hiányzik egy végpont.

helytelen argumentum

A `gcd` függvény argumentuma negatív vagy nulla.

helytelen parancslista adatkérés

A program elérte a command függvényt, de nem tudja végrehajtani, mert egy másik függvény aktív; esetleg a parancsértelmező nincs teljesen inicializálva. Ez előfordulhat az *acad.lsp*, *acadr13.lsp* vagy egy *.mnl* fájlból történő `command` függvényhíváskor.

illegális típus a bal oldalon

Az *.lsp* fájl nem tisztán ASCII karakterekből áll, de egy szövegszerkesztő program elmentette, és formatáló kódokat is tartalmaz.

input megszakadt

Hiba, vagy idő előtti "fájl vége" jel észlelése a fájl beolvasásának megszakadását okozza.

ismételt belépés az AutoLISP programba nem lehetséges

Az AutoCAD/AutoLISP kommunikációs puffert egy éppen aktív függvény használja. Amíg ennek végrehajtása be nem fejeződik, nem hívható meg új függvény.

kifejezés nem értékelhető ki

A hibát okozhatja a tizedespont helytelen elhelyezése, vagy más hibás formátumú kifejezés.

konzol megszakítás

A függvény működése közben a felhasználó megnyomta a Ctrl+C billentyűket.

lebegőpontos hiba

(Csak UNIX rendszereknél.) Az operációs rendszer hibát észlelt a lebegőpontos számításban.

LISPSTACK túlcsordulás

Az AutoLISP veremtárolója megtelt. Ezt okozhatja túl sok rekurzív függvényhívás, vagy nagyon hosszú argumentumlista.

maximális karakterlánc-hossz túllépése

A függvénynek átadott karakterlánc több mint 132 karakterből áll.

nem megfelelő objektum a függvényben

A **vmon** függvénylapozó hibásan szerkesztett függvényt észlelt.

(file) nem nyitható meg inputra -- LOAD sikertelen

A **load** függvényhívásban megnevezett fájl nem található, vagy a felhasználó nem jogosult a fájl olvasására.

nullával való osztás

A nullával való osztás nem megengedett.

osztási túlcsordulás

Nagyon kis számmal való osztás érvénytelen hányadost eredményezett.

quit/exit megszakítás

A **quit** vagy az **exit** függvény meghívásának az eredménye.

rossz asszociációs lista

Az **assoc** függvényt olyan listával látta el, amely nem kulcsérték-listákból áll.

rossz ENTMOD lista

Az **entmod** függvénynek átadott argumentum nem érvényes rajzelem-adatlista (nem olyan, ahogy azt az **entget** függvény visszaadja).

rossz ENTMOD listaérték

Az **entmod** függvénynek átadott asszociációs listában található allisták egyike hibás értéket tartalmaz.

rossz formális argumentumlista

A függvény kiértékelésekor az AutoLISP a formális argumentumok listáját nem találta érvényesnek. Lehetséges, hogy nem is függvényről, hanem adatlistáról van szó.

rossz függvény

A lista első eleme nem érvényes függvénynév. Talán egy változó neve, vagy egy szám. Az üzenet azt is jelentheti, hogy a függvény helytelenül van definiálva. (Ne felejtse el, hogy a formális argumentumok listájára mindenképpen szükség van.)

rossz funkciókód

Rossz függvényazonosítót adott át a **tablet** parancsnak.

rossz grvecs listaérték

Olyasvalami szerepel a **grvecs** függvénynek átadott listában, ami nem két- vagy háromdimenziós pont.

rossz grvecs mátrixérték

A **grvecs** függvénynek átadott mátrix formátuma hibás, vagy rossz adattípust (például STR, SYM stb.) tartalmaz.

rossz konverziós kód

Érvénytelen térazonosítót adott át a **trans** függvénynek.

rossz lista

Helytelenül kialakított listát adott át a függvénynek. A hibát az is okozhatja, ha egy valós szám tizedesponttal kezdődik. Ilyen esetekben mindig írja ki a kezdő nullát.

rossz node

A **type** függvény érvénytelen típusú tétellel találkozott.

rossz pontargumentum
rossz pontérték

Helytelenül definiált pontot (két valós számból álló listát) adott át a függvénynek. Ügyeljen arra, hogy ne kezdjen egy valós számot tizedesponttal; ilyen esetekben írja ki a kezdő nullát is.

rossz pontokból álló lista

Az F, MP vagy AP opcióhoz mellékelt lista üres, vagy nem pontokat tartalmaz. Az **ssget** és a **grvecs** függvény jelez ilyen hibát.

rossz ssget lista

Az (**ssget** "**X**") függvényhívásban átadott argumentum nem érvényes adatlista (nem olyan, ahogy azt az **entget** függvény visszaadja).

rossz ssget listaérték

Az (**ssget** "**X**") asszociációs listájában található allisták egyike hibás értéket tartalmaz.

rossz ssget mód karakterlánc

A hibát az okozza, hogy az **ssget** függvénynek a *mode* argumentumban átadott karakterlánc érvénytelen.

rossz típusú argumentum

Hibás típusú argumentumot adott át a függvénynek. (Például nem alkalmazhatja a **strlen** függvényt egy egész számra.)

rossz típusú node a listában

A **foreach** függvény érvénytelen típusú tétellel találkozott.

rossz valós szám észlelése

Érvénytelen valós számot próbált átadni az AutoLISP nyelvből az AutoCAD programnak.

rossz xdata lista

A hibát az okozza, hogy az **xdsize, ssget, entmod, entmake** vagy **txtbox** függvénynek átadott bővített rajzelem-adatlista formátuma hibás.

rosszul elhelyezett tizedespont

Egy valós számot tizedesponttal kezdett. Ilyen esetekben mindig írja ki a kezdő nullát is.

rosszul kialakított karakterlánc

Egy fájlból beolvasott karakterlánc idő előtt véget ért.

rosszul kialakított lista

Egy fájlból beolvasott lista idő előtt véget ért. A leggyakoribb ok a nyitó és befejező zárójelek, illetve idézőjelek párosításának hibája.

túl hosszú karakterlánc

A **setvar** függvénynek átadott karakterlánc túl hosszú.

túl kevés argumentum

Egy beépített függvénynek túl kevés argumentumot adott át.

túl kevés argumentum a grvecs számára

Nem adott át elegendő argumentumot a **grvecs** függvénynek.

túl sok argumentum

Egy beépített függvénynek túl sok argumentumot adott át.

üres függvény

Egy üres definícióval rendelkező függvény kiértékelésére irányuló kísérlet.

Belső hibák

Az alábbi hibaüzenetek az AutoLISP program belső hibáira utalnak. Amennyiben az alábbi üzenetek valamelyikével találkozik, kérjük írja meg az AutoDesknek egy Hibajelentés (Bug Report) űrlapon.

busz hiba

(Csak UNIX rendszereknél.) Az operációs rendszer busz hibát észlelt.

érvénytelen utasítás

(Csak UNIX rendszereknél.) Az operációs rendszer nem megengedett gépi utasítást észlelt.

felfüggesztés

(Csak UNIX rendszereknél.) Az operációs rendszer felfüggesztési (*hangup*) jelzést észlelt.

nem várt jelzés nnn

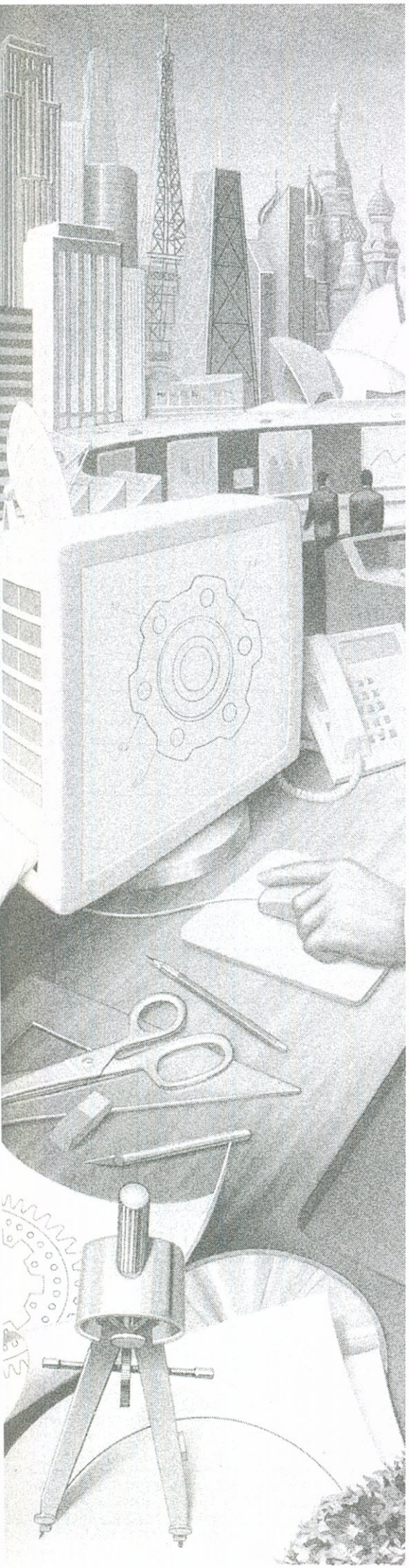
(Csak UNIX rendszereknél.) Az operációs rendszertől nem várt jelzés érkezett.

rendszerhívás rossz argumentummal

(Csak UNIX rendszereknél.) Az operációs rendszer az AutoLISP által generált hibás rendszerhívást észlelt.

szegmenshatár megsértése

(Csak UNIX rendszereknél.) Az operációs rendszer a műveleti memóriaterületen túli címzést észlelt.



III. RÉSZ

PROGRAMOZHATÓ PÁRBESZÉDPANEL REFERENCIA

TARTALOM

III. RÉSZ PROGRAMOZHATÓ PÁRBESZÉDPANEL REFERENCIA 615

18. FEJEZET ■ PROGRAMOZHATÓ PÁRBESZÉDPANELEK 617

Párbeszédpanelek alkotórészei 618
A DCL fájl szerkezete 620
A *base.dcl* és *az acad.dcl* fájl 621
DCL fájlok hivatkozásai 621
A DCL szintaxisa 622
Zónadefiníciók 622
Zónahivatkozások 623
Attribútumok és attribútumértékek 624
Megjegyzések 624
DCL hibakezelés 624
DCL fájlok szemantikai ellenőrzése 625
Egy minta párbeszédpanel 627
DCL technikák 629
Zónák elosztása egy csoportban 629
Zónák közti üres hely 630
Üres hely a jobb oldalon vagy alul 630
Üres hely egy bekeretezett sor vagy oszlop körül 631
A kilépő nyomógomb szövegének megváltoztatása 632
Irányelvek a tervezéshez 634
A platformmal kapcsolatos szempontok 634
Esztétikum és ergonómia 634
Következetes kialakítás és tiszta nyelvezet 635
Felhasználói szabadság 635
Feledhető hibák 636

Súgó szolgáltatás 637
Hátrányos helyzetű felhasználók 637
Szín 637
Súgó 638
Kezelés a billentyűzetről 638
Érthetőség és egyszerűség 638
Nagy kezdőbetűs írásmód 638
Párbeszédpanelek, területek és oszlopok címei 638
Kezelőszervek címkéi 638
Promptok és üzenetek 638
Rövidítések 639
Elrendezés 639
Méret és elhelyezés 639
Letiltott zónák 639
Párbeszédpanelek egymásba ágyazása 640
Párbeszédpanelek elrejtése 640
Alapértelmezés szerinti értékek felajánlása 640
Adatbevitel billentyűzetről 641
Nyelvi megfontolások 641
Az előre definiált zónákra és csoportokra vonatkozó irányelvek 642
Nyomógombok 642
Csoportok 643
Szerkesztőablakok 643
Ábra-nyomógomb és ábrazóna 644
Listaablakok 644
Rádiógombok, rádiósorok és -oszlopok 644
Csúszkák 645
Szöveg 645
Kapcsolók 645
Hibakezelés 646
Hibazónák 646
Figyelmeztető ablakok 646
Kapcsolódó dokumentáció 647

19. FEJEZET ■ A PÁRBESZÉDVEZÉRLŐ NYELV 649

Attribútumok 650
Tiltott attribútumok 651
Előre definiált attribútumok 651
action 654
alignment 654
allow_accept 654
aspect_ratio 655
big_increment 655
children_alignment 655

children_fixed_height 656
children_fixed_width 656
color 656
edit_limit 657
edit_width 657
fixed_height 658
fixed_width 658
height 658
initial_focus 659
is_bold 659
is_cancel 659
is_default 659
is_enabled 660
is_tab_stop 660
key 660
label 660
layout 661
list 661
max_value 661
min_value 662
mnemonic 662
multiple_select 663
small_increment 663
tabs 663
value 664
width 664
Felhasználó által definiált attribútumok 664
A DCL zónák funkciók szerinti áttekintése 665
Előre definiált aktív zónák 665
Zónacsoportok 666
Dekoratív és tájékoztató jellegű zónák 667
Szövegcsoportok 667
Párbeszédpanelek kilépő nyomógombjai és hibazóna 667
Tiltott zónák 668
DCL zónák katalógusa 668
bekeretezett oszlop 668
bekeretezett rádióoszlop 669
bekeretezett rádiósor 669
bekeretezett sor 670
nyomógomb 670
oszlop 671
szöveglánc 671
párbeszéd 672
szerkesztőablak 672
hibazóna 673

ábra 673
ábra-nyomógomb 674
listaablak 674
ok 675
ok, mégsem 675
ok, mégsem, súgó 675
ok, mégsem, súgó, hibazóna 676
ok, mégsem, súgó, info 676
bekezdés 676
legördülő lista 677
rádiógomb 678
rádióoszlop 679
rádiósor 679
sor 680
csúszka 680
szöveg 681
szövegrész 682
kapcsoló 682
helykitöltő (spacer) 682
helykitöltő (spacer_0) 683
helykitöltő (spacer_1) 683

III. RÉSZ
PROGRAMOZHATÓ PÁRBESZÉDPANEL REFERENCIA

A *Programozható párbeszédpanel referencia* bemutatja, hogyan tervezzen és alkalmazzon, az AutoCAD által használtakhoz hasonló párbeszédpaneleket. Az AutoLISP és az ADS egyaránt biztosít olyan függvényeket, amelyek párbeszédpaneleket jelenítenek meg, felhasználói adatbevitelt értékelnek ki és így tovább.

Mivel a párbeszédpaneleket AutoLISP és ADS alkalmazások vezérlik, ismernie kell a 10., "Párbeszédpanelek kezelése" című fejezetben, illetve az *AutoCAD Devolopers Guide* című kézikönyv 6., "Programming Dialog Boxes" című fejezetében leírt párbeszédpanel-kezelő függvényeket. A 18.,"Programozható párbeszédpanelek" című fejezet AutoLISP függvényeket használ a példáiban. A 11., "AutoLISP gyakorló példa" című fejezet bemutatja, hogyan adjon párbeszédpanel felületet egy alkalmazáshoz.

A beépített és a felhasználó által definiált párbeszédpanelek AutoCAD általi támogatása függ a platformtól. Az alkotóelemek és a működés változatlan marad, a megjelenés azonban a platform grafikus felhasználói felületétől (GUI) függ. A PDB szolgálatás automatikus elrendezést és méretezést biztosít, amellyel az Ön munkáját minimalizálja és segíti a következetes tervezést .

18

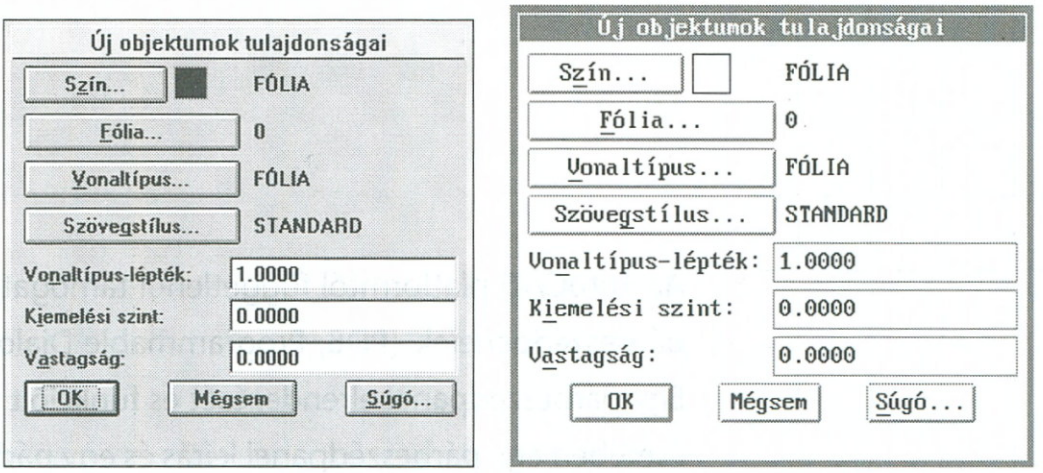
PROGRAMOZHATÓ PÁRBESZÉDPANELEK

Az AutoCAD platformtól függetlenül támogatja a programozható párbeszédpenelek (PDB, Programmable Dialogue Box) használatát. Egy párbeszédpanel elrendezését és funkcióit valamennyi platform esetében egy párbeszédpanel leírás és egy párbeszédpanel alkalmazás definiálja. A párbeszédpanel tényleges megjelenése az egyes platformok grafikus felhasználói felületének (GUI, Graphical User Interface) függvénye. A képi megjelenésben mutatkozó bármilyen különbség a platformokhoz tartozó megjelenítő eszközmeghajtók különbözőségéből adódik.

A párbeszédpanelek definiálása Párbeszédvezérlő nyelven (Dialog Control Language, DCL) írt ASCII fájlok útján történik. A párbeszédpanelben elhelyezkedő elemek (zónák) például, nyomógombok, szerkesztőablakok elhelyezkedését azok DCL-fájlbeli sorrendje határozza meg. Minden elem méretét és funkcióját a zóna attribútuma vezérli. A párbeszédpanelek kialakítása szigorúan szabályozott: a párbeszédpanel méretének és alkotóelemei elrendezésének beállítása automatikusan történik, minimális pozicionálási információ felhasználásával. Az alkotórészek meghatározzák a párbeszédpanel viselkedését; a párbeszédpanel használata és viselkedése ugyanakkor függ az alkalmazástól, amely felhasználja. Az AutoLISP és az ADS rendelkezik a párbeszédpanelek vezérlésére szolgáló függvényekkel.

Ez a fejezet a párbeszédpanel alkotóelemeit ismerteti. Elmagyarázza továbbá a DCL fájl szerkezetét és szintaxisát, valamint AutoLISP és DCL példaprogramokat mutat be a párbeszédpanelek használatára. Ebben a fejezetben talál néhány olyan DCL programozási technikát is, amelyekkel kezelhetők az elrendezési problémák.

A következő ábra azt mutatja, hogyan jelenik meg egy párbeszédpanel különböző platformokon, ugyanazt a DCL leírást használva:

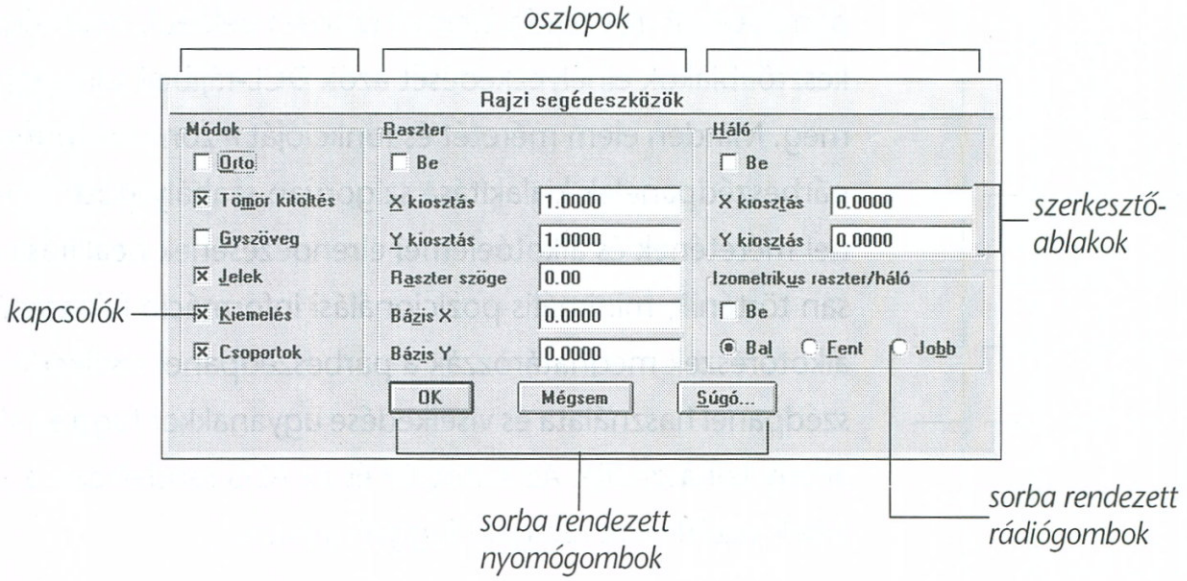


AutoCAD, Windows rendszer alatt *AutoCAD, DOS rendszer alatt*

Párbeszédpanel megjelenése különböző platformokon

Párbeszédpanelek alkotórészei

Az alábbi ábra az AutoCAD szabvány párbeszédpaneljeinek egyikét mutatja be, egyes alkotórészeinek megnevezésével. Ezeket az alkotóelemeket a párbeszédpanelek létrehozása és adaptálása során *zónáknak* nevezzük



Egy párbeszédpanel jellegzetes zónái

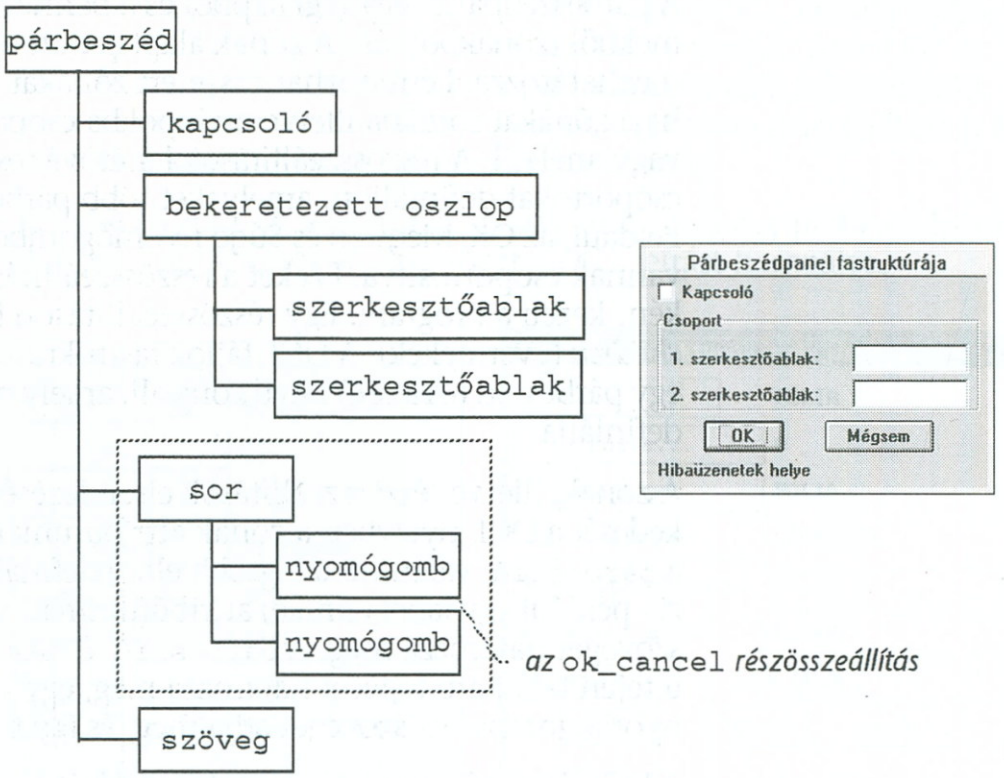
A párbeszédpanel egy téglalapból és a benne elhelyezkedő alkotóelemekből (zónákból) áll. A zónák alaptípusait a PDB szolgáltatás előre meghatározza. Létrehozhat összetett zónákat (részösszeállításokat) is, ha a zónákat sorokba illetve oszlopokba csoportosítja, bekeretezve vagy anélkül. A részösszeállítások lehetővé teszik, hogy olyan zónacsoportokat definiáljon, amelyeket több párbeszédben is használhat. Például, az OK, Mégsem és Súgó nyomógombok egy részösszeállításba vannak csoportosítva. Ezeket a részösszeállításokat egyszerű zónákként kezeli a program. Egy részösszeállításon belüli zónák neve *children* (gyermekek). A DCL fájlok fastruktúrát alkotnak. A fa csúcsán egy párbeszéd (`dialog`) nevű zóna áll, amely magát a párbeszédpanelt definiálja.

A zónák, illetve részösszeállítások elrendezését, megjelenését és viselkedését a DCL nyelvben a zónák attribútumai határozzák meg. Maga a `párbeszéd`, valamint a legtöbb előre definált zónatípus is rendelkezik például egy label (címke) attribútummal, amely a zónához tartozó szöveget határozza meg. A `párbeszéd címkéje` a párbeszédpanel tetején található fejlécet határozza meg, egy nyomógomb `címkéje` a nyomógombra írt szöveget adja meg, és így tovább.

A DCL új zónák (*prototípusok*) definiálását is megengedi, melyek nem feltétlenül egy párbeszédpanelhez kapcsolódnak. Ez akkor hasznos, ha ugyanazt az alkotóelemet több párbeszédpanelben is használni kívánja. A prototípusokra hivatkozhat más DCL fájlokból és ugyanúgy megváltoztathatja attribútumaikat, mint az előre definiált zónák esetében.

Egy párbeszédpanel alapvetően zónák hierarchiája, a DCL fájl pedig ennek olvasható változata -- olvasható mind a fejlesztők, mind az AutoCAD PDB szolgáltatás számára. Egy DCL fájl hierarchiáját annak szintaxisa jelzi. A hierarchia szemléltetését a DCL fájl tabulálásával érheti el. Javasoljuk, hogy Ön is a jelen fejezet példáiban alkalmazott tabulálási stílust használja.

A DCL használatával a következő ábrán szemléltetett párbeszédpanelleírást hozhat létre. A jobb oldalon látható párbeszédpanel jelzi a zónák helyét a hierarchiában.



Egy párbeszédpanel fastruktúrája

A párbeszédpanelek DCL leírása ASCII fájlokban történik, nagyon hasonlóan a programozási nyelvek forráslistáihoz vagy az AutoCAD menü (*.mnu*) forrásszintű leírásához. A DCL fájlok kiterjesztése *.dcl*. Egyetlen *.dcl* fájl tartalmazhatja egy vagy több párbeszédpanel leírását, vagy csak prototípus zónákat és részösszeállításokat más *.dcl* fájlokban történő felhasználás céljára.

Mielőtt hozzáfog egy párbeszédpanel programjának megírásához és a hibakereséshez, tervezze meg részletesen mind a párbeszédpanelt, mind az alkalmazást. A bevitt adatok sorrendje minden felhasználó esetében változni fog. Ez kevésbé lineáris programstruktúrát követel, mint a hagyományosabb programok, viszont tükrözi az egyes felhasználók munkamódszerét.

A DCL fájl szerkezete

A párbeszédpanelek definiálásán kívül a DCL fájlok más DCL fájl által használt prototípusokat és részösszeállításokat is defniálhatnak, valamint más DCL fájlokban található prototípusokra és részösszeállításokra is hivatkozhatnak. Egy DCL fájl az alábbi három részből áll. Ezek sorrendje tetszőleges. Az alkalmazásától függően csak egy, vagy több részre van szükség.

- *Hivatkozások más DCL fájlokra.* Ezek *include direktívákból* állnak, mint azt a "DCL fájlok hivatkozásai" című résznél a 621. oldalon ismertetjük.

- *Prototípus zónák és részösszeállítások definíciói.* Ezek zónadefiníciók, amelyekre egymást követő zónadefiníciókban (beleértve a párbeszédpanel definíciókat) hivatkozhat.
- *Párbeszédpanel-definíciók.* Ezek nagyon hasonlóak az előző részben bemutatott példához.

A *base.dcl* és *az acad.dcl* fájl

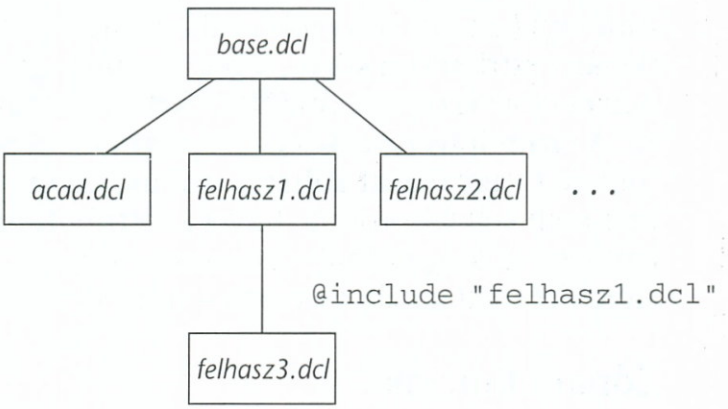
A *base.dcl* és az *acad.dcl* fájl az AutoCAD része, és az AutoCAD *support* könyvtárában található.

A *base.dcl* fájlban az alapvető, előre definiált zónák és zónatípusok, valamint az általánosan használt prototípusok definíciói találhatók. A PDB szolgáltatás teszi lehetővé az előre definiált zónák átdefiniálását. Az *acad.dcl* fájl az AutoCAD által használt összes párbeszédpanel definícióját tartalmazza.

Figyelmeztetés A *base.dcl* fájlban előforduló bármely hiba az eredeti AutoCAD párbeszédpanelek, valamint a saját és más alkalmazások által használt egyéni párbeszédpanelek megjelenését is tönkreteheti.

DCL fájlok hivatkozásai

Amikor párbeszédpaneleket készít, akkor egy új, az alkalmazáshoz illő DCL fájlt kell létrehoznia. Az összes DCL fájl használhatja a *base.dcl* fájlban definiált zónákat. Egy DCL fájl másik DCL fájlban definiált zónákat is használhat, ha a másik fájlt egy *include direktívában* nevezi meg. A DCL fájlokból saját hierarchiát állíthat fel, mint az a következő ábrán látható:



Egy DCL fájl hierarchiája

A fenti ábrán látható *felhasz1.dcl* és *felhasz2.dcl* fájlok egymástól függetlenek, de a *felhasz3.dcl* a *felhasz1.dcl* fájlban definiált zónákat használja. Az include direktíva alakja a következő:

```
@include fájlnév
```

ahol a *fájlnév* egy idézőjelek között álló karakterlánc, amely a másik DCL fájl teljes nevét tartalmazza (a *.dcl* kiterjesztéssel együtt). A következő direktíva például a *felhmag.dcl* nevű fájlra hivatkozik:

```
@include "felhmag.dcl"
```

Amennyiben Ön csak a fájlnevet adja meg, akkor a PDB szolgáltatás először az aktuális könyvtárban keresi a fájlt, azután (az include direktívát tartalmazó) DCL fájllal azonos könyvtárban. Ha a fájl nevét a teljes elérési útvonallal együtt adja meg, akkor a PDB csak a megadott könyvtárban keresi a fájlt.

Megjegyzés Az Ön által készített DCL fájlok nem használhatják az *acad.dcl* fájlban definiált párbeszédpaneleket. Nem írhatja: `@include "acad.dcl"`. Ha azonban hasonló párbeszédpanelt kíván készíteni, átmásolhatja a definíciókat saját DCL fájljába.

A DCL szintaxisa

Ebben a részben a zónáknak, azok attribútumainak és az attribútumok értékének a meghatározására szolgáló DCL szintaxisát ismertetjük. Új zónák *definícióval* hozhatók létre. Ha a zónadefiníció egy párbeszédpanelen kívül szerepel, akkor az egy prototípus vagy egy részösszeállítás. Prototípusok és részösszeállítások zónára történő *hivatkozások* segítségével használhatók a párbeszédpanelek definícióiban. Egy definícióra vonatkozó valamennyi hivatkozás örökli az eredeti zóna attribútumait. A prototípusok hivatkozásaiban ezeknek az örökölt attribútumoknak az értékét megváltoztathatja, vagy új attribútumokat hozhat létre. A részösszeállítások hivatkozásaiban nem változtathatja meg, és nem egészítheti ki az attribútumokat.

Amennyiben egy zóna több változatára van szüksége, amelyekben néhány attribútum közös, legegyszerűbb egy olyan prototípust definiálni és elnevezni, amely csak ezeket a közös attribútumokat tartalmazza. A prototípus hivatkozásaiban azután megváltoztathatja az attribútumokat vagy újakat adhat hozzájuk, közös attribútumokat azonban már nem kell felsorolnia. Mivel az attribútumok öröklődnek, gyakrabban lesz szüksége zónahivatkozások létrehozására—különösen az előre definiált zónák esetében—mint új zónák definiálására.

Zónadefiníciók

A zónadefiníciók alakja a következő:

```
név : tétel1 [ : tétel2 : tétel3 ... ] {
    attribútum = érték;
    ...
}
```

ahol minden *tétel* egy előre definiált zóna. Az új zóna (*név*) örökli az összes megadott tétel (*tétel1*, *tétel2*, *tétel3*, ...) attribútumait, a kapcsos zárójeleken (`{}`) belüli attribútum-definíciók pedig kiegészí-

tik, vagy helyettesítik (ha az attribútumnév azonos) az örölkött definíciókat. Amikor a definíciónak több "szülője" van, az attribútumok balról jobbra haladva élveznek elsőbbséget. Más szóval, ha egynél több tétel tartalmazza ugyanazt az attribútumot, akkor a sorban előrébb álló tétel kerül felhasználásra.

Ha az új definíció nem tartalmaz gyermeket, akkor prototípus, és hivatkozásaiban megváltoztathatók vagy bővíthetők az attribútumai. Amennyiben gyermekekkel rendelkező részösszeállítás, attribútumai nem változtathatók meg.

A zóna, illetve a prototípus zóna *neve* csak betűket, számokat vagy az aláhúzás karaktert (_) tartalmazhatja, és betűvel kell kezdődnie. A zónanevek különbséget tesznek a kis- és nagybetűk között: a `nagygomb` például nem ugyanaz, mint a `NagyGomb` vagy a `NAGYGOMB`. Ügyeljen a nagybetűk használatára.

Egy nyomógomb (belső) definíciója a következő:

```
button : tile {
   fixed_height = true;
   is_tab_stop = true;
}
```

A *base.dcl* fájlban a `default_button` definíciója a következő:

```
default_button : button {
   is_default = true;
}
```

A `default_button` örökli a `button` zóna `fixed_height` és `is_tab_stop` attribútumait. Hozzáad egy új `is_default` nevű attribútumot, melyet `true` értékre állít be.

Zónahivatkozások

A zónahívatkozás alakja a következő.

```
név;
```

vagy

```
: név {
     attribútum = érték;
     . . .
}
```

ahol a *név* egy előzőleg definiált zóna neve. A zónanevek különbséget tesznek a kis- és nagybetűk között. Az első esetben a hivatkozás a *név* nevű zónában definiált összes attribútumot átveszi. A második esetben a kapcsos zárójelen belül attribútum-definíciók kiegészítik vagy helyettesítik a *név* zónától örökölt definíciókat. Mivel ez egy zónahivatkozás, nem pedig definíció, az attribútumok megváltoztatása a zónának csak erre a felhasználására vonatkozik.

Megjegyzés A második alak csak a prototípusokra vonatkozik, a részösszeállításokra nem.

Egy párbeszédpanel definíciójában a `spacer` (helykitöltő) zóna az elrendezés miatt szerepel. Egyedi attribútumai nincsenek, így a következő alakban lehet rá hivatkozni:

```
spacer;
```

Az `ok_cancel` zóna egy, a *base.dcl* fájlban definiált részösszeállítás, ezért csak az első alak szerint lehet rá hivatkozni:

```
ok_cancel;
```

Egy önálló zóna attribútumai viszont átdefiniálhatók. Például létrehozhat egy nyomógombot ugyanazokkal a tulajdonságokkal, de más szöveggel, ha az alábbi formát használja:

```
: retirement_button {
  label = "Viszlát";
}
```

Bővebb információ a "A kilépő nyomógomb szövegének megváltoztatása" című résznél a 632. oldalon található.

Attribútumok és attribútumértékek

Egy zónadefiníció vagy -hivatkozás kapcsos zárójelein belül a következő alakban adhat meg attribútumokat és rendelhet hozzájuk értéket:

```
attribútum = érték ;
```

ahol az *attribútum* egy érvényes kulcsszó, az *érték* pedig az attribútumhoz rendelt érték. Az attribútumot az értékétől egy egyenlőségjel (=) választja el, az értékadást pedig egy pontosvessző (;) zárja le.

A zónanevekhez hasonlóan az attribútumnevek és -értékek is különbséget tesznek a kis- és nagybetűk között. A `Szélesség` és a `szélesség` nem ugyanaz; a `True` és a `true` sem adja ugyanazt az eredményt.

Megjegyzések

A megjegyzések általában információszolgáltatás céljából kerülnek a programkódba. Ez az információ lehet a program vagy a kapcsolódó fájlok neve, a szerző neve, illetve bármilyen használatra vonatkozó vagy magyarázó információ. A DCL fájlban a megjegyzéseket két törtjel (//) előzi meg. Bármi álljon is a // jelek és a sor vége között, azt a PDB szolgáltatás nem veszi figyelembe. A DCL a C nyelv szabályai szerinti megjegyzéseket is megengedi. Ennek alakja: `/* megjegyzés szövege */`. A nyitó `/*` és a záró `*/` különböző sorokban is állhat.

DCL hibakezelés

A PDB szolgáltatás a DCL fájlt az első betöltés alkalmával ellenőrzi. Amennyiben az AutoCAD szintaktikai hibával, az attribútumok

helytelen használatával vagy egyéb hibával találkozik (például egy aktív zóna kulcsértékének hibás megadása), akkor nem tölti be a DCL fájlt, és a hibák listáját egy *acad.dce* nevű szövegfájlba írja. Az *acad.dce* fájl tartalmából megtudhatja, hogy mi volt a hiba. Amikor az AutoCAD sikeresen beolvasott egy DCL fájlt, törli az *acad.dce* fájlt. Az AutoCAD az *acad.dce* fájlt az aktuális munkakönyvtárban helyezi el.

Amennyiben az alkalmazás több DCL fájlt használ, az AutoCAD minden újabb DCL fájl betöltésekor felülírja az *acad.dce* fájlt (vagy törli, ha nincs hiba). A program kipróbálásakor így az *acad.dce* csak az utolsó DCL fájlból származó hibákat mutatja (ha voltak ilyenek). Ebből következően célszerű a hibák felderítéséhez minden DCL fájlt a **load_dialog** függvénnyel betölteni. Az alábbi **load_dialog** függvény betölti a *hellofjl.dcl* fájlt:

```
Parancs:    (load_dialog "hellofjl")
3
```

A **load_dialog** függvény a DCL azonosító számmal tér vissza. Jegyezze meg ezt a számot, mert ez adódik át argumentumként a következő **new_dialog** és **unload_dialog** függvényhívásoknak.

Amennyiben a fellépő hibák következtében az AutoCAD az *acad.dce* fájlba ír, egy figyelmeztető ablak jelenik meg. Egyébként a **load_dialog** egy pozitív egész számmal tér vissza, amely a DCL fájlt azonosítja. Ezt az értéket átadhatja a **new_dialog** függvénynek a fájlban szereplő egyedi párbeszédpanelek inicializálása céljából.

Egyes hibákról a **new_dialog** futása közben megjelenő figyelmeztető ablak tudósítja, de ezek nem kerülnek feljegyzésre az *acad.dce* fájlban. Sikeres végrehajtás esetén a **new_dialog** függvény `T`, máskülönben `nil` értékkel tér vissza. Amennyiben a **new_dialog** visszatérési értéke `T`, használja a **start_dialog** függvényt a párbeszéd megjelenítésére.

Figyelmeztetés Az AutoCAD program leállhat, ha Ön meghívja a **start_dialog** függvényt, miután a **new_dialog** függvény `nil` értékkel tért vissza. Egyes platformokon egy **new_dialog** hívás anélkül is aktivizálná a párbeszédpanelt, hogy elindítaná azt. Ez szintén leállíthatja az AutoCAD futását. Ilyen platformokon a **new_dialog** és a **start_dialog** függvényeket interaktív módon csak egy másik függvénybe foglalva hívhatja meg. Például:

```
(if (new_dialog "hello" 3)(setq retcode (start_dialog)))
```

Ha egyenként kiszűrte a hibákat valamennyi DCL fájlból, betöltheti a programot, és kipróbálhatja a párbeszédpaneleket együtt is.

DCL fájlok szemantikai ellenőrzése

Az AutoCAD a szemantikai ellenőrzés (auditálás) négy szintje (0–3) közül biztosít választási lehetőséget a DCL fájlok esetében. Ez olyan kódokat keres a DCL fájlban, amelyek valószínűleg hibásak vagy

szükségtelenek. Ez az ellenőrzés a DCL fájl betöltése közben történik. Egy DCL fájl ellenőrzési szintjének beállítására helyezzen el egy, az alábbihoz hasonló sort bárhol a fájlban (de ne egy zónadefiníción belül):

```
dcl_settings : default_dcl_settings { audit_level = 3; }
```

Amennyiben az Ön DCL fájlja hivatkozik bármely másik DCL fájlra (az `@include` direktíva segítségével), akkor a `dcl_settings` sort csak egy fájlba írja be. A megadott ellenőrzési szintet fogja használni a többi fájl is.

A szemantikai ellenőrzés szintjei

Szint	*Leírás*
0	Nincs ellenőrzés. Csak akkor használja, ha a DCL fájlok ellenőrzése már megtörtént, és azóta azokban nem történt változás.
1	Hibák. Olyan hibákat keres, amelyek az AutoCAD leállását eredményezik. Ez az alapértelmezés szerinti szint, és szinte alig okoz késlekedést. A hibák között szerepelhet nem definiált zónák és visszatérő prototípus-definíciók használata.
2	Figyelmeztetés. Olyan DCL hibákat keres, amelyek a párbeszédpanelek nemkívánatos elrendezését vagy viselkedését eredményezik. Egy módosított DCL fájlt legalább egyszer ellenőrizni kell ezen a szinten. A figyelmeztető szint olyan hibákat talál meg, mint a szükséges attribútum hiánya vagy nem megfelelő attribútumérték.
3	Javaslatok. Felesleges attribútum definíciókat keres

Az ellenőrzési szolgáltatás legjobb kihasználása érdekében a fejlesztők az `audit_level` értékét 3-ra állítsák a fejlesztés során, és csak akkor töröljék ki az `dcl_settings` sort, mikor átadják a fájlokat a felhasználóknak .

Egy minta párbeszédpanel

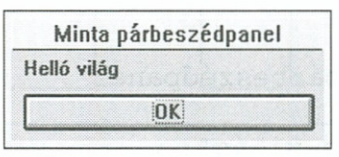
Egy párbeszédpanel megjelenítésére és a benne levő zónák vezérlésére az AutoLISP vagy az ADS porgramok használhatók. Az alábbi AutoLISP alkalmazás (melyet a "Párbeszédpanelek kezelése" című, 10. fejezetben fejtettünk ki és magyaráztuk el részletesen) az ebben a részben található párbeszédpanelek meghívására szolgál. Az alábbi AutoLISP függvénydefiníciót beírhatja a parancsprompt után, vagy elmentheti egy *.lsp* fájlba és a **load** függvény segítségével töltheti be.

```
(defun c:hello ()
  (setq dcl_id (load_dialog "hello.dcl"))
  (if (not (new_dialog "hello" dcl_id) ) (exit))
  (action_tile "accept" "(done_dialog)")
  (start_dialog)
  (unload_dialog dcl_id)
  (princ)
)
```

Készítsen egy *hello.dcl* nevű ASCII szövegfájlt, amely tartalmazza a következő párbeszédpanel-leírást:

```
hello : dialog {
     label = "Minta párbeszédpanel";
     : text {
       label = "Helló világ";
     }
     : button {
       key = "accept";
       label = "OK";
       is_default = true;
     }
}
```

Feltéve, hogy a **C:HELLO** függvény definiálva van, ez a DCL programrészlet az alábbi párbeszédpanelt eredményezi, ha a Parancs prompt után a **hello** szót írja be.



A minta párbeszédpanel

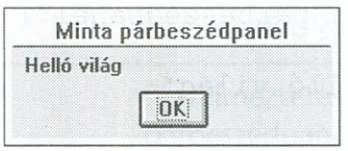
A párbeszédpanel címe "Minta párbeszédpanel", és csak egy Helló világ üzenetet, illetve egy OK nyomógombot tartalmaz. A párbeszédpanel mérete és elrendezése az alapértelmezésből adódott. Van azonban egy kis probléma: az OK nyomógomb a párbeszédpanel teljes szélességén végighúzódik.

A párbeszédpanel megjelenésének javítása érdekében módosíthatja a DCL fájlt, és két attribútummal egészítheti ki a zónát. Annak megakadályozására, hogy a nyomógomb teljesen kitöltse a rendelkezésre álló

helyet, adjon meg egy `fixed_width` attribútumot `true` értékkel. Ennek következtében a nyomógomb kerete összeszűkül annyira, hogy alig lesz szélesebb, mint a benne lévő szöveg. A nyomógomb középre igazítása céljából adjon meg egy `alignment` attribútumot `centered` értékkel. Az alapértelmezés szerint a zónák balra igazítva helyezkednek el egy oszlopban. A DCL leírás ezek után a következő:

```
hello : dialog {
    label = "Minta párbeszédpanel";
    : text {
      label = "Helló világ";
    }
    : button {
      key = "accept";
      label = "OK";
      is_default = true;
      fixed_width = true;
      alignment = centered;
    }
}
```

A párbeszédpanel most így jelenik meg:



A minta párbeszédpanel javított elrendezéssel

Bár a nyomógombok jó lehetőséget kínálnak az attribútumok bemutatására, léteznek saját párbeszédpanelekben használható, szabvány kilépő nyomógomb részösszeállítások is. További információ található a "Párbeszédpanelek kilépő nyomógombjai és hibazóna" című részнél a 667. oldalon. Ha a fenti ábrán bemutatottal gyakorlatilag azonos párbeszédpanelt kíván létrehozni, használja az `ok_only` részösszeállítást, mint azt a következő példa mutatja :

```
hello : dialog {
  label = "Minta párbeszédpanel";
  : text {
    label = "Helló világ";
  }
  ok_only;
}
```

DCL technikák

Az alábbiakban leírt technikák segítségével számos elrendezéssel kapcsolatos probléma megoldható.

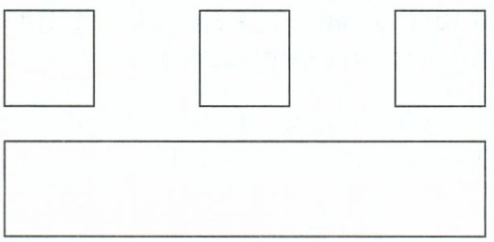
Döntse el, hogy az alapértelmezés szerinti elrendezés megfelel-e az Ön által elkészített párbeszédpanel számára. Amennyiben probléma merül fel, állítsa át az elrendezést a csoport szinten (az alapértelmezések). Különálló zónákat csak akkor állítson át, ha feltétlenül szükséges.

Zónák elosztása egy csoportban

Amikor a zónákat elrendezi egy párbeszédpanelben, az egyes zónák relatív méretei szerint sorokba és oszlopokba kell rendeznie őket. A következő DCL egy három zónából álló sort definiál, amely egy másik zóna fölött, annak teljes hosszában helyezkedik el:

```
: column {
   : row {
      : compact_tile {
      }
      : compact_tile {
      }
      : compact_tile {
      }
   }
   : large_tile {
   }
}
```

Amennyiben a `compact_tile` nevű alkotóelemek szélessége rögzített (`fixed_width`) és a `large_tile` zóna szélesebb, mint a fölötte lévő, `compact_tile` zónákból álló sor által igényelt minimális tér, akkor az alapértelmezés szerint ez az összeállítás a következőképpen fog megjelenni:



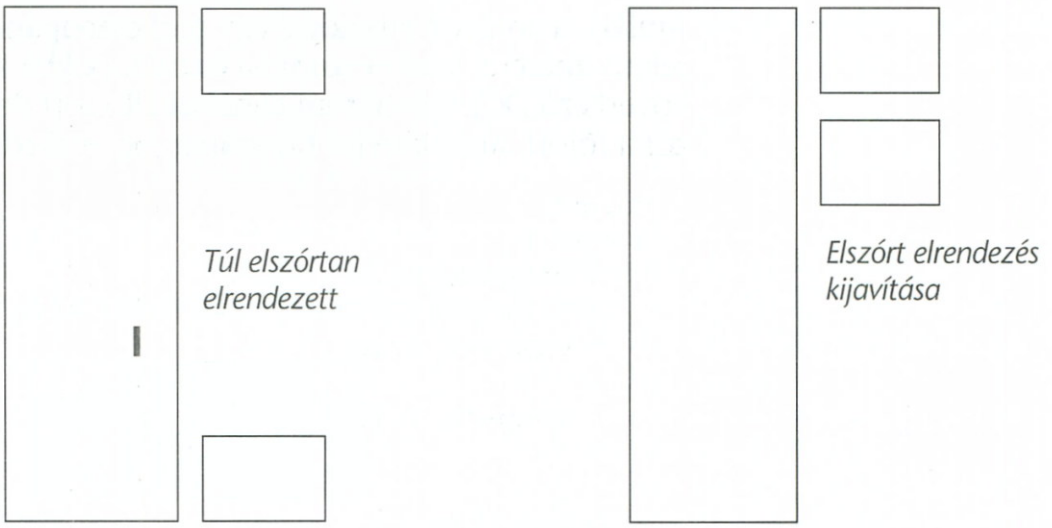
Alapértelmezés szerinti vízszintes elrendezés

A sorban az első `compact_tile` zóna bal széle a `large_tile` zóna bal szélével, az utolsó `compact_tile` zóna jobb széle pedig a `large_tile` zóna jobb szélével esik egy vonalba. Ezen belül a zónák eloszlása egyenletes. Oszlopok találkozása esetén hasonló a helyzet.

Az alapértelmezés szerinti elrendezést a `spacer_0` és `spacer_1` zónák használatával befolyásolhatja, amelyek a *base.dcl* fájlban definiált `spacer` zóna változatai.

Zónák közti üres hely

Amennyiben jelentős különbség van két, szomszédos oszlop zónái által elfoglalt hely között, a kevesebb helyet igénylő oszlop zónái túl elszórtan helyezkedhetnek el. A `fixed_height` attribútum `true` értékre állításával javítható a megjelenésük. Az eredményt a következő diagram mutatja:



Függőleges zónaelrendezés

Üres hely a jobb oldalon vagy alul

Előfordulhat, hogy egy párbeszédpanel jobb oldalán kihasználatlan tér mutatkozik. Ezt néha egy olyan `text` zóna okozza, amelynek explicit módon megadott szélessége nagyobb, mint az aktuális értéke által megkívánt szélesség. Ez gyakran elfogadható; ha azonban problémát jelent, a text zónát egy kerettel veheti körül a `boxed_row` (bekeretezett sor) zóna segítségével.

Példaként következzék egy részlet az AutoCAD Rajzelemkészítési módok párbeszédpanel leírásából:

```
: text {
    key = "l_text";
    width = 18;
    fixed_width = true;
}
```

Ez a `text` zóna semmit sem jelenít meg (`value` attribútumának értéke üres karakterlánc) mindaddig, amíg az alkalmazás nem helyez el benne valamilyen szöveget egy ehhez hasonló utasítással:

```
(set_tile "l_text" "Fólia")
```

vagy:

```
ads_set_tile(current_dlg, "l_text", "Fólia");
```

Mivel a `"Fólia"` nem igényli mind a 18 karakternyi helyet, ezért a párbeszédpanel jobb szélén fölösleges szabad tér mutatkozik.

Hasonló helyzet áll elő, amikor egy `errtile` zónát (további információ a "Párbeszédpanelek kilépő nyomógombjai és hibazóna" című résznél a 667. oldalon) használ a hibaüzenetek megjelenítésére. Hacsak nem látható éppen egy hibaüzenet, úgy tűnik, mintha fölösleges szabad tér lenne a párbeszédpanel alján. Ebben az esetben egy, a párbeszédpanel felső részén elhelyezett külön `spacer` zóna segíthet egyenletesebbé tenni a függőleges elrendezést.

Üres hely egy bekeretezett sor vagy oszlop körül

Ha egy bekeretezett sor vagy oszlop label (címke) attribútuma csupán egy szóközt (`" "`) tartalmaz vagy üres (`""`), a keret megjelenik ugyan a csoport körül, de semmilyen szöveg nem látható. Az egyetlen szóközből álló címke nem jelenik meg szóközként az ablakban. A szóköz értékű és az üres címkék elrendezésének módjában azonban mutatkozik némi különbség:

- Amennyiben a `label` egyetlen szóközből áll, akkor az a terület, amelyet a szöveg függőleges irányban a kereten *belül* elfoglalt, elvész, de az a terület, amelyet a címke a keret *felett* elfoglalt, megmarad. Mivel a szöveg aktuális elhelyezése a bekeretezett sorban és oszlopban platformonként változik, az üres hely variációk platformtól függőek.
- Amennyiben a címke üres karakterlánc, a függőleges terület elvész mind a keret felett, mind azon belül. Ez nem függ a platformtól.

Az alábbi DCL kód esetében az első két oszlopot körülvevő keretek felső vonala garantáltan egy vonalba esik (*Y* irányú helyzetük azonos), a harmadik oszlopot körülvevő keret esetében pedig biztos, hogy sem a felső vonala felett, sem az alatt nincs üres hely az alapértelmezés szerinti szegélyen kívül:

```
: row {
   : boxed_column {
      label = "Valamilyen szöveg";
   }
   : boxed_column {
      label = " ";          // egyetlen szóköz: alapért. sz.
   }
   : boxed_column {
      label = "";           // üres karakterlánc
   }
}
```

A kilépő nyomógomb szövegének megváltoztatása

Egyes párbeszépanelek esetén előfordulhat, hogy meg kívánja változtatni valamelyik kilépő nyomógomb szövegét. Ha például egy olyan párbeszédpanelt készít, amellyel adatok törölhetők, akkor biztonságosabb az OK helyett egy Rombol nevű nyomógombot alkalmaznia. Ennek elérésére használja a `retirement_button` prototípust az alábbi módon:

```
rombol_gomb : retirement_button {
     label = "&Rombol";
     key = "rombol";
}
```

Figyelje meg az *és* jel (&) használatát a label attribútumban. Ez egy mnemonikot rendel a fájlhoz. Ebben az esetben a R betű lesz aláhúzva a címke nyomógombon, és ez lesz a mnemonik.

Megjegyzés Amikor megváltoztat egy meglévő kilépő nyomógomb részösszeállítást, a *base.dcl* fájlból szerezze meg a megfelelő DCL kódot, és ne a kézikönyvből, ugyanis a *base.dcl* fájl az Ön által használt rendszer esetében eltérő lehet.

Miután definiált egy saját kilépő nyomógombot, bele kell azt foglalnia egy részösszeállításba, amely megjelenésében és működésében megfelel a szabvány csoportoknak. A következő példa bemutatja az `ok_cancel_help` nyomógomb aktuális definícióját.

```
ok_cancel_help : column {
   : row {
     fixed_width = true;
     alignment = centered;
     ok_button;
     : spacer { width = 2; }
     cancel_button;
     : spacer { width = 2; }
     help_button;
   }
}
```

Hozzon létre egy új részösszeállítást, amelyben az `ok_button` helyett az új nyomógomb szerepel:

```
destroy_cancel_help : column {
  : row {
    fixed_width = true;
    alignment = centered;
    destroy_button;
    : spacer { width = 2; }
    cancel_button;
    : spacer { width = 2; }
    help_button;
  }
}
```

A szabvány részösszeállításban az OK az alapértelmezés szerinti nyomógomb, de ezt az attribútumot nem adtuk hozzá a `rombol_gomb`-hoz. Amikor a párbeszédpanel akciója romboló hatású (vagy nagyon időigényes), javasoljuk, hogy a Mégsem nyomógombot tegye meg az alapértelmezés szerinti nyomógombnak. Ebben az esetben alapértelmezés szerinti és megszakító nyomógombként is működik:

```
rombol_mégsem_súgó : column {
  : row {
    fixed_width = true;
    alignment = centered;
    rombol_gomb;
    : spacer { width = 2; }
    : cancel_button { is_default = true; }
    : spacer { width = 2; }
    help_button;
  }
}
```

Mivel megváltoztattunk egy attribútumot, az eredeti Mégsem nyomógombot prototípusként kell használni. A kettőspont most a `cancel_button` nyomógomb előtt jelenik meg.

Figyelmeztetés Amennyiben a Mégsem nyomógomb és az alapértelmezés szerinti nyomógomb azonos (az `is_default` és az `is_cancel` attribútum értéke egyaránt `true`) és Ön elmulaszt valamely másik nyomógombhoz olyan akciót rendelni, amely meghívja a **`done_dialog`** függvényt, akkor egyetlen más nyomógombbal sem lehet kilépni a párbeszédpanelből, így azt *mindig* csak elvetni tudja.

Irányelvek a tervezéshez

Ahhoz, hogy egy párbeszédpanelt jól tervezzen meg, nemcsak annak gyakorlati célját kell szem előtt tartania, hanem a párbeszédpanel megjelenésének esztétikai, valamint használatának ergonómiai szempontjait, továbbá nem utolsó sorban annak a platformnak (vagy platformoknak) a GUI szabványait is, amelyen az alkalmazást futtatni kívánja. Az alábbi rész néhány vezérelvet szolgáltat a GUI és a párbeszédpanel tervezéséhez, a hordozhatósághoz, valamint az előre definiált zónákhoz és csoportokhoz.

A platformmal kapcsolatos szempontok

Az AutoCAD PDB szolgáltatása nagyrészt független a platformtól; vannak azonban kivételek (további információ a "Rádiógombok, rádiósorok és -oszlopok" című résznél a 644. oldalon). Egyetlen DCL fájl is használható párbeszédpanelt generál az összes olyan platformon, amely támogatja az AutoCAD programot, a párbeszédpanel megjelenése pedig alkalmazkodik az adott platform megjelenítési szabványaihoz. Ezek a szabványok azonban eltérőek. Egyes grafikus felhasználói felületek például a szerkesztőablakot egy négyzet alakú keretben jelenítik meg; más felületek viszont csak a szerkesztősort jelenítik meg. Hasonlítsa össze párbeszédpanel-tervét minden támogatott platform szabványával, és győződjön meg róla, hogy minden környezetben ésszerű.

Minden grafikus felhasználói felületnek vannak olyan jellemzői, amelyekkel a PDB szolgáltatás nem rendelkezik. Ha ezeket a jellemzőket kívánja használni, akkor olyan párbeszédpaneleket kell kifejlesztenie, amelyek az adott platform szolgáltatásait használják; az ilyen párbeszédpaneleket azonban nem lehet változtatás nélkül átvinni más AutoCAD platformra.

Mivel a billentyűzet mnemonikokat nagyon széles körben használják, ezeket meg kell adnia még akkor is, ha egyes platformok nem ismerik fel őket.

Esztétikum és ergonómia

Fontos a párbeszédpanel jó megjelenése. Ha a panel látványa zavaros, akkor nehéz használni, és hatékonysága is kisebb. A zónákat is az ergonómiai szempontokat figyelembe véve kell elhelyezni, azért, hogy könnyen lehessen őket használni. Vegye figyelembe, hogy melyek lesznek a leggyakrabban használt zónák, tegye őket feltűnő helyre, és úgy rendezze el azokat, hogy könnyen lehessen köztük mozogni, különösen mikor egymással összefüggően használják őket.

Következetes kialakítás és tiszta nyelvezet

Egy alkalmazás felhasználói felülete rendelkezzen következetes belső felépítéssel, és álljon összhangban a kapcsolódó alkalmazásokkal is. Egy ismeretlen párbeszédpanel kezelését könnyebb megtanulni, ha az összhangban van az adott alkalmazás, más kapcsolódó alkalmazások, illetve az operációs rendszer többi párbeszédpaneljével. Ennek egyik példája az olyan nyomógombok következetes elhelyezése, mint az OK és a Mégsem. A zónák egyes fajtáinak kezelésében—abban, hogy miként lehet egy szöveges ablakba írni, vagy hogyan választható ki egy lista valamelyik tétele—szintén következetességre kell törekedni. Az összhang elérésének legjobb módja, ha ugyanazokat a zónákat használja fel, az azokat kezelő programrészekkel együtt. A szabványosítás hozzájárul a következetesség érvényesítéséhez.

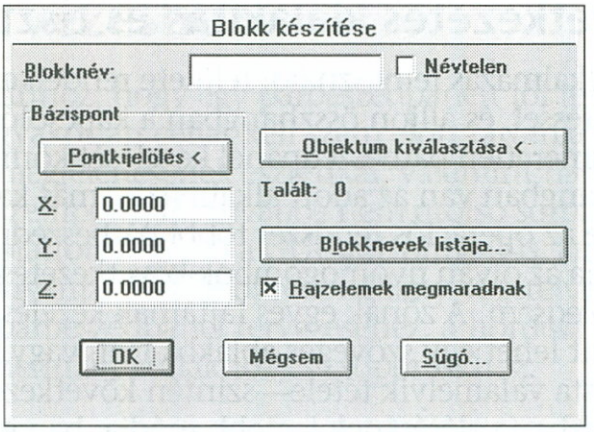
Használjon szabványos definíciókat a párbeszédpanel vezérléséhez. Ez csökkenti az elvégzendő munkát, hozzájárul a következetesség érvényesítéséhez, és könnyebbé teszi a felhasználóknak a párbeszédpanelek megismerését és használatát.

Használjon tiszta nyelvezetet. Bár a párbeszédpanelek a grafikus felület részének tekinthetők, a zónák és az általuk nyújtott információ szöveges jellegű. A párbeszépanelben alkalmazott címkék, a nyomógombok nevei, az üzenetekben előforduló kifejezések legyenek lényegretörőek és egyértelműek. Kerülje a zsargont és az olyan műszaki kifejezéseket, amelyeket a felhasználó esetleg nem ért.

Felhasználói szabadság

Biztosítson a felhasználónak bizonyos szabadságot abban, hogy miként adja meg a párbeszédpanelben az adatokat. A párbeszédpanelek egyik előnye a parancssor útján történő kapcsolattartáshoz képest az, hogy a felhasználók nincsenek a pormptok szigorú sorrendjéhez kötve. Egy párbeszédpanelben a felhasználó bármilyen sorrendben megadhatja az adatokat. Bizonyos kötöttségekre szükség lehet—egy opció kiválasztása például egy másik elhalványítását vonja maga után—de csak olyan korlátokat állítson fel, amelyre az alkalmazás működése alapos okot szolgáltat.

A következő ábrán például a Blokk készítése párbeszédpanel látható. Ennek a párbeszédpanelnek a segítségével a felhasználó először kijelölheti a bázispontot, majd kiválaszthatja az objektumokat, megadhatja a blokk nevét; ezután meggondolhatja magát és kiválaszthat más objektumokat, végezetül megnyomja az OK nyomógombot. Az objektumokat esetleg még a bázispont kijelölése előtt kiválaszthatja. A legfőbb korlátozás logikus: a felhasználó nem adhat meg blokknevet, ha a Névtelen kapcsolót előzőleg bekapcsolta.

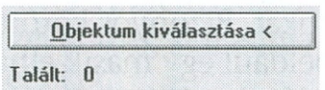


A Blokk készítése párbeszédpanel

Több párbeszédpanel esetén azok inkább egymás tetején jelenjenek meg ahelyett, hogy kilépésre szólítaná fel a felhasználót ahhoz, hogy egy másikat meg tudjon hívni. Mindig tegye lehetővé a felhasználó számára, hogy visszatérhessen az először megjelenített párbeszédpanelhez. Ez nem kényszeríti választásra, mielőtt elhagyja a párbeszédpanelt. Mivel az aktuális párbeszédpanel az előző tetején jelenik meg, ez emlékezteti a felhasználót az összefüggésre: honnan jött, és hová fog visszatérni.

Valahányszor a felhasználó megváltoztat valamilyen állapotot vagy opciót, kapjon azonnali visszajelzést. Ha a felhasználó kiválaszt valamit, azt nyomban tegye láthatóvá vagy írja le. Amennyiben egy választás kizár más választási lehetőségeket, az utóbbiakat rögtön halványítsa el.

Az AutoCAD Szín kiválasztása párbeszédpaneljében például egy ábrában látható a kiválasztott számnak megfelelő szín. A Blokk készítése minta párbeszédpanelben a kiválasztott rajzelemek számáról mindig tájékoztat egy üzenet az Objektum kiválasztása nyomógomb alatt:



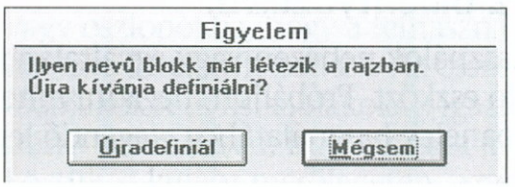
Állapot üzenet megjelenítésére használt szövegzóna

Feledhető hibák

A párbeszédpanelek legyenek "megbocsátók", így a felhasználó szabadon kísérletezhet, s nem kell tartania attól, hogy visszafordíthatatlan hibát követ el. A kisebb hibákat jelezze a párbeszédpanel alsó részében, a hibazónában megjelenő üzenetekkel. A komolyabb hibák jelzésére alkalmazzon figyelmeztető ablakot. Az `alert` függvény egy egyszerű figyelmeztető ablakot jelenít meg (egyetlen OK nyomógombbal). További információ az `"alert"` függvényről a 13. fejezetben, az `"ads_alert"` függvényről pedig az *AutoCAD Devolopers Guide* című kézikönyv 3. fejezetében található.

Amennyiben a felhasználó romboló vagy időigényes akciót választ ki, a párbeszédpanel jelenítsen meg egy figyelmeztető ablakot, amely választási lehetőséget biztosít a felhasználónak: folytatja a műveletet, vagy elveti azt.

A Blokk készítése párbeszédpanelben például egy figyelmeztető ablak jelenik meg, ha a felhasználó már létező objektumot kíván létrehozni. A felhasználó ekkor választhat, hogy folytatja és felülírja az eredeti blokkot, vagy elveti a műveletet anélkül, hogy változást idézett volna elő.



Figyelmeztető ablak folytató és megszakító opcióval

A felhasználót figyelmeztető beágyazott párbeszédpanelek térjenek vissza az előző párbeszédpanelhez. Csak súlyos vagy potenciálisan végzetes hiba esetén szakítsa meg az egymásba ágyazott párbeszédpanelek láncolatát.

Súgó szolgáltatás

A súgónak a felhasználó rendelkezésére kell állnia. A felhasználónak biztosított segítség mértéke attól függ, mennyire bonyolult az alkalmazás, és mennyire megától értetődő egy-egy párbeszédpanel. Ajánlatos elhelyezni egy olyan Súgó nyomógombot az alkalmazás fő párbeszédpaneljében, amely a fontos információt közlő párbeszédpanelt jeleníti meg. A Súgó nyomógomb legtöbb esetben az AutoLISP **help** függvényével meghívott AutoCAD szabvány Súgó párbeszédpaneljének megjelenítésére szolgál.

Bonyolultabb alkalmazás esetén, vegye fontolóra egy olyan helyzetérzékeny súgó kifejlesztését, amelyben minden párbeszépanelhez külön súgó tartozik .

Hátrányos helyzetű felhasználók

A hátrányos helyzetű felhasználókkal kapcsolatos megfontolások mindenki számára megkönnyítik a program használatát. Párbeszédpaneljeinek tervezésekor vegye figyelembe a következőket:

Szín

Sokan nem tudnak különbséget tenni bizonyos színek között. Ha színeket használ valamilyen információ közlésére, kiegészítésképpen jelenítse azt meg más módon is (általában szöveggel). Amennyiben lehetséges használja a célplatformban az AutoCAD által alkalmazott színsémát.

Az AutoCAD színek kiválasztására szolgáló párbeszédpanelje például egy szöveges üzenetben is kiírja a szín nevét vagy számát, és ábrában is megjeleníti a színt.

Súgó

Sok felhasználónak nehézségei vannak a kézikönyv apró betűivel, esetleg fizikailag képtelen a könyvek használatára. Már egyetlen, a fő párbeszédpanelben elhelyezett Súgó nyomógomb is sokat segíthet.

Kezelés a billentyűzetről

Egyes felhasználók nehezen vagy egyáltalán nem tudják használni a pozícionáló eszközt. Próbáljon megadni mnemonikokat, hogy a párbeszédpanelek használatához elegendő legyen a billentyűzet is.

Érthetőség és egyszerűség

A gondosan tervezett párbeszédpanelek világos és egyszerű nyelvezetükkel segíthetnek a csökkent verbális felfogóképességű felhasználókon. Ne kívánja, hogy a felhasználók sok különböző dolgot tartsanak észben. Ehelyett törekedjék következetes szóhasználatra, és ahol lehet, kínáljon fel választékot.

Nagy kezdőbetűs írásmód

Az alábbiakban néhány általános irányelv következik a párbeszédpanelekben látható szövegek nagy kezdőbetűs írásmódjával kapcsolatban:

Párbeszédpanelek, területek és oszlopok címei

Használja a "szalagcímes" nagybetűs írásmódot: kezdjen nagybetűvel minden szót az elsőtől az utolsóig, kivéve a névelőket, elöljárószavakat, kötőszavakat. Ha azonban a párbeszédpanelt egy menüből hívta meg (nem a Parancs promptból), akkor a címnek meg kell egyezni a menüponttal.

Kezelőszervek címkéi

Használja a "szalagcímes" nagybetűs írásmódot a vezérlő zónák, például nyomógombok esetében. A címke után *ne* tegyen pontot. Szöveges ablakok vagy legördülő listák címkéi után tegyen kettőspontot (:). Használhatja a mondatokra jellemző nagybetűs írásmódot is (amelyben csak az első betűt és azokat a betűket írja naggyal, amelyeket egyébként is azzal írna), ha a címke hosszú vagy kérdést tartalmaz.

Promptok és üzenetek

Használja a mondatokra jellemző nagybetűs írásmódot.

Rövidítések

Kerülje a rövidítéseket. Félreérthetők vagy nehezen értelmezhetők lehetnek. Amennyiben helyszűke miatt rövidítenie kell a kifejezéseket, próbáljon meg egy csoporton (például bekeretezett oszlopon) belül következetesen rövidíteni . Legyen következetes. *Ne* írjon ki egyes szavakat teljesen, másokat pedig rövidítve.

Elrendezés

A párbeszédpanel logikailag együvé tartozó részeit rendezze sorokba vagy oszlopokba, hogy a felhasználó balról jobbra vagy felülről lefelé haladva áttekinthesse azt. Az összetartozó mezőket (például szerkesztőablakokat és listaablakokat) igazítsa egymáshoz mind függőlegesen, mind vízszintesen, hogy amikor a felhasználó a Tab billentyűvel vált köztük, a kurzor merőlegesen, egyenes vonalban mozduljon el.

Amennyiben az adatok természetes sorrendben megadhatók — például az *X*, *Y* és *Z* koordináták esetében—, akkor a mezőket aszerint rendezze el. A bekeretezett területeket igazítsa egy vonalba mind függőlegesen, mind vízszintesen . Ne hagyjon túl sok "fehér foltot" a bekeretezett területek körül vagy azok között. Szükség esetén növelje meg a szélességüket jobb felé.

Méret és elhelyezés

Az információ világos megjelenítése érdekében a párbeszédpanel ne legyen nagyobb a szükségesnél.

Megjegyzés Az IBM VGA az AutoCAD által jelenleg támogatott legalacsonyabb felbontású monitor. Győződjön meg róla, hogy a párbeszédpanel elfér-e ezen a monitoron.

Alapértelmezés szerint az AutoCAD első ízben a grafikus képernyő közepén helyezi el a párbeszédpeneleket. Egyes platformok lehetővé teszik, hogy a párbeszédpanelek a felhasználó által utoljára definiált helyen jelenjenek meg. Amennyiben az alkalmazást ilyen platformhoz fejleszti, a következő alkalommal a párbeszédpanelt a felhasználó által kívánt helyen nyissa meg. Az AutoLISP esetében a **`new_dialog`** és a **`done_dialog`** függvények gondoskodnak a párbeszédpanelek elhelyezéséről. Az ADS esetében az **`ads_done_positioned_dialog()`** és az **`ads_new_positioned_di-alog()`** függvények teszik ezt.

Letiltott zónák

Amennyiben egy zóna vagy terület az opciók aktuális beállításából adódóan érvénytelen vagy közömbös, azonnal tiltsa le azt. Ezáltal a zóna vagy terület elhalványul, és a felhasználó nem tudja kiválasztani. Ha lehet, ne éljen túl gyakran a zónák letiltásának a lehetőségével. Túl sok elhalványított zóna zavaró lehet.

Amennyiben egy zóna valamilyen értéket tartalmaz, a letiltása ne befolyásolja az adott értéket: engedélyezése után mutassa ugyanazt az értéket. A "varázslatos" módon változó értékek bosszantó többletmunkát okoznak a felhasználónak.

Párbeszédpanelek egymásba ágyazása

Az AutoCAD legfeljebb nyolc párbeszédpanel egymásba ágyazását engedi meg. Az egymásba ágyazott párbeszédpanelek kezeléséről szóló információt a "Párbeszédpanelek egymásba ágyazása" című résznél a 313. oldalon találhatja meg. Illesszen három pontot (...) annak nyomógombnak a címkéjébe, amely a beágyazott párbeszédpanelt jeleníti meg, kivéve, ha az egy figyelmeztető ablak. Háromszoros mélységnél jobban *ne* ágyazzon egymásba párbeszédpaneleket. A négyszeres beágyazás akkor ésszerű, ha a negyedik szinten csak figyelmeztető ablakok vannak. Mivel a párbeszédpanelek első alkalommal a képernyő közepén jelennek meg, a beágyazott panelt a fő párbeszédpanelnél kisebbre készítse.

Párbeszédpanelek elrejtése

Amennyiben a felhasználónak a grafikus képernyőn kell valamit kiválasztania mielőtt a párbeszédpanel bezáródott volna, akkor a párbeszédpanelnek ideiglenesen be kell záródnia, hogy a felhasználó láthassa a képernyőt és elvégezhesse a kiválasztást. Miután a kiválasztás megtörtént, a párbeszédpanelt újra meg kell nyitnia. Ezt nevezzük a párbeszédpanel *elrejtésének* .

A párbeszédpanel elrejtését kiváltó nyomógomb címkéje *ne* tartalmazzon három pontot. Használjon ehelyett egy szóközt és egy azt követő kisebb jelet (<) a címkében. Amikor a párbeszédpanel elrejti önmagát, egy promptnak kell megjelennie, amely elmagyarázza a felhasználónak, hogy mit kell tennie.

Az adatbevitelt a legtöbb esetben a **getxxx** függvények egyikével kérheti. Ezek a függvények rendelkeznek egy olyan argumentummal, amelyben Ön megadhatja a promptot.

A párbeszédpanel újbóli megjelenésekor tartalmazzon valamilyen visszajelzést a kiválasztás megtörténtéről. Ez lehet egy új információ a szerkesztőablak mezőjében, egy felfrissített listaablak, a státuszt jelző szöveges üzenet, vagy ezek kombinációja.

Alapértelmezés szerinti értékek felajánlása

Ajánljon fel ésszerű alapértelmezés szerinti értékeket az összes adatbevitel és opció számára. A helyesen megválasztott alapértelmezések hozzásegítik a felhasználót a párbeszédpanel gyors és könnyű kitöltéséhez.

Célszerű az alapértelmezés szerinti értékeket aktualizálni —más szóval megjegyezni a felhasználó által előzőleg beállított értékeket és a továbbiakban azokat tekinteni alapértelmezés szerinti értékeknek—a párbeszédpanel minden egyes használatakor. Még ha a felhasználónak meg is kell változtatnia néhányat ezek közül, ez kevesebb munkával jár, mintha minden egyes alkalommal elölről kezdené az egészet.

Adatbevitel billentyűzetről

Amikor párbeszédpanelt készít, meghatározhatja a billentyűzetről bevitt rövidítések kezelését. Nem minden platform támogatja a mnemonik billentyűk használatát, azonban néhány ezek közül lehetővé teszi, hogy a felhasználó a Tab billentyű segítségével léptesse a fókuszt. Néhány billentyűzetről bevitt rövidítés az összes párbeszédpanelben közös. A Tab billentyű például általában lehetővé teszi a felhasználó számára, hogy zónáról zónára lépkedjen, a Spacebar billentyűvel pedig a kapcsolókat lehet ki- és bekapcsolni. Minden aktív zóna egy tabulátor helyet jelentsen (ez az alapértelmezés).

Két *gyorsító* billentyű ismeretes. Az *elfogadás* billentyűje (rendszerint a ↵ billentyű) elfogadja a párbeszédpanelt és a beírt értékeket. A *törlés* billentyűje (egyes platformokon az Esc, másokon a Ctrl+C) elveti a párbeszédpanelt és a bevitt értékeket. Amikor az AutoCAD először jeleníti meg a párbeszédpanelt, az egyik zónán található a billentyűzet kezdeti fókusza. A felhasználó által beírtak ezt a zónát befolyásolják egészen addig, amíg a felhasználó egy másik zónára nem irányítja a fókuszt.

A felhasználó egyik zónáról a másikra léphet a Tab billentyűvel, egy másik zóna kiválasztásával vagy egy billentyűzeti rövidítés (*mnemonik*) megadásával. A Rajzi segédeszközök párbeszédpanelben például kiválaszthatja az Izometrikus raszter/Háló rádiógombok egyikét az L, a T vagy az R billentyű lenyomásával. Egyes platformok megkívánják, hogy a mnemonikot az Alt billentyűvel együtt nyomja le. A zónák közti mozgás megváltoztatja a fókuszt, de nem hajt végre kiválasztást. Ha a felhasználó azt a zónát kívánja kiválasztani, amelyiken a fókusz van, meg kell nyomnia az elfogadó billentyűt. Egyes típusú zónáknál a dupla kattintás az elfogadó billentyű lenyomásával egyenlő.

Nyelvi megfontolások

A különböző nyelvek szavai gyakran több karaktert igényelnek, mint azok angol megfelelői. Amennyiben a párbeszédpanel címkéit más nyelvre fordítják, hagyjon annyi helyet a terjeszkedésre, amennyit

csak lehet. Az alábbi táblázat néhány ismert AutoCAD kifejezést mutat be azok angol és német megfelelőivel.

AutoCAD kifejezések megfelelői magyarul, angolul és németül

magyar	*angol*	*német*
Vonal	Line	Linie
Ív	Arc	Bogen
Kör	Circle	Kreis
3D Vonalláncok	3D Polylines	3D-Polylinien
Átmérő-méretezés	Diameter dimensioning	Durchmesserbemaßung
Fóliák	Layers	Layer
Vonaltípusok	Linetypes	Linientypen
Rajzelemkészítési kódok	Entity creation modes	Modus für Objekterzeugung
Válasszon objektumot	Select objects	Objekte wählen
OK	OK	OK
Mégsem	Cancel	Abbruch
Súgó	Help	Hilfe

Az előre definiált zónákra és csoportokra vonatkozó irányelvek

Ebben a részben az előre definiált zónák és csoportok egyes fajtáival kapcsolatban javasolt tervezési irányelveket soroljuk fel.

Nyomógombok

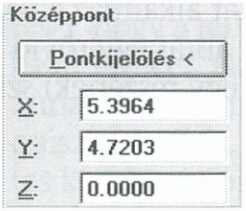
A nyomógombhoz rendelt akció látható legyen a felhasználó számára és azonnal következzen be. A nyomógomb címkéje legyen félreérthetetlen. Rendszerint egy igének kell lennie, amely a gomb lenyomásának hatását fejezi ki, bár más címke—például az OK vagy Opciók—szintén elfogadható, ha jelentése világos. Az olyan párbeszédpanelek esetében, amelyek egy másikat hívnak vagy az aktuálisat elrejtik, további információ található a "Párbeszédpanelek egymásba ágyazása" című résznél a 640. oldalon és a "Párbeszédpanelek elrejtése" című résznél a 640. oldalon.

Az egy oszlopban lévő nyomógomboknak azonos szélességűeknek kell lenniük. Más esetekben a nyomógomboknak rögzített szélességűeknek kell lenniük (`fixed_width = true;`, vagy `children_fixed_width = true;`) a közös szülői csoportban.

Csoportok

A bekeretezett csoport (sor vagy oszlop) neve *csoportablak* vagy *terület*. Egy terület a felhasználó számára is látható módon különíti el és nevezi meg az összetartozó kezelőszerveket. A terület annyi zónát, sort és oszlopot tartalmazhat (keret nélkül), amennyi szükséges. A terület címkéje jelezze annak célját.

Az egymással összefüggő kezelőszerveket egy területen belül helyezze el. A Bázispont csoport a Blokk készítése minta párbeszédpanelben jól példázza ezt a technikát:



Címkével és kerettel rendelkező csoportból képzett terület

Ezzel együtt ne vigye túlzásba a területek használatát. A szabad tér is hatásos módja a zónák csoportosításának. *Ne* keretezzen be egy listaablakot. Ez túlságosan zsúfoltnak hat.

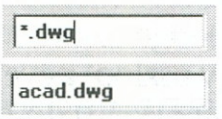
Szerkesztőablakok

A szöveg beírására szolgáló mező hossza hozzávetőleg egyezzen meg egy átlagos adatbeviteli mező hosszával. Ha bizonytalan, valós számok esetében vegyen 10, szövegek esetében 20 karakternyi hosszúságot.

A szerkesztőablak címkéje végződjön kettőspontra (:).

Amennyiben valamilyen korlátozás vonatkozik a felhasználó által beírható adatra, helyezzen el egy olyan szövegzónát a szerkesztőablaktól jobbra, amely röviden ismerteti ezeket a korlátozásokat. Amennyiben például a felhasználónak egy fájlnevet kell beírnia, nem szükséges elmagyarázni, hogy mi az. Ha azonban a karakterlánc egy szám, amely nem haladhatja meg a 100-at, akkor célszerű emlékeztetni erre a korlátozásra.

Adatok -- például pontok -- számára jobb két vagy három szerkesztőablakot biztosítani, mintsem elvárni, hogy a felhasználó emlékezzen egy pont megadásának parancssorban szokásos formájára. Kivételt képezhet az olyan különleges formátumú adat bevitelére szánt szerkesztőablak, mint például az, amelyik az AutoCAD fájlkiválasztó párbeszédpaneljében a globális karaktereket tartalmazó minta megadására szolgál.



Szerkesztőablak különleges formátumú adat számára

Ábra-nyomógomb és ábrazóna

Amennyiben a felhasználó figyelmeztetésére egy ábra-nyomógombot vagy ábrazónát használ ikonként—például egy stoptáblát—használja azt következetesen minden párbeszédpanelben.

Amikor ábra-nyomógombokat alkalmaz a választási lehetőségek bemutatására, egészítse ki az ábrát egy rövid szöveges ismertetővel, különösen, ha az ábra (vagy egy részének) színe szerepet játszik a kiválasztásában.

Listaablakok

Mivel a listák vízszintes irányban nem görgethetők, a listaablak szélessége a lista leghosszabb tételéhez alkalmazkodjon. Biztosítson egy címkét (vagy szövegzónát) a lista tartalmának magyarázatára, hacsak nem a listaablak a párbeszédpanel legfőbb zónája. Ebben az esetben a párbeszédpanel címe is elegendő—a listaablaknak azonban mindenképpen kell címkét adnia, ha azt akarja, hogy a felhasználó egy mnemonik segítségével elérhesse azt.

A lista tételeit szedje betűrendbe, hacsak nincs ésszerű oka, hogy ettől eltérjen. Amennyiben a lista hossza rögzített és viszonylag rövid, fontolja meg egy rádióoszlop használatát a listaablak helyett.

Rádiógombok, rádiósorok és -oszlopok

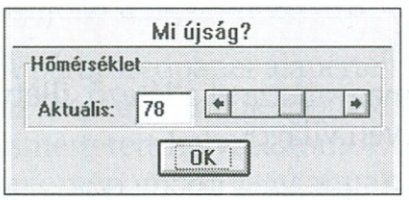
A rádióoszlopok tetszetősebbek és egyszerűbben használhatók, mint a rádiósorok. A rádiósorok csak akkor megfelelőek, ha kevés számú (általában kettő - négy) nyomógombot tartalmaznak, illetve ha a címkék rövidek.

Amennyiben egy máshol kiválasztott opció a rádiósor vagy -oszlop választási lehetőségeit érvénytelenné illetve feleslegessé teszi, halványítsa el az egész sort vagy oszlopot. Néhány esetben előfordulhat, hogy egy máshol kiválasztott opció a rádiósor vagy -oszlop bizonyos választási lehetőségeit érvénytelenné illetve feleslegessé teszi. Ilyen helyzetekben a nyomógombokat külön-külön is el lehet halványítani. Egyes grafikus felület-szabványok elfogadják ezt, mások azonban nem. Amennyiben olyan platformra fejleszt, amely ezt nem engedi meg, csoportosítsa más módon a választási lehetőségeket—például a rádiósor vagy -oszlop kettébontásával, hogy a két részt külön lehessen letiltani.

Csúszkák

A csúszka "felbontóképessége" ne legyen túl durva. Ha a csúszkához például csak öt diszkrét érték tartozik, és a párbeszédpanelben öt centimétert foglal el, akkor a felhasználónak egy centiméterrel kell el mozdítania ahhoz, hogy lássa a változást. Kerülje az ugrálást a csúszka méretének arányos beállításával.

Amennyiben a felhasználónak tudnia kell a csúszkával beállított értéket, a párbeszédpanel jelenítse meg a csúszka aktuális értékét is. Aktualizálja ezt az értéket, valahányszor a csúszka elmozdul. Javasoljuk, hogy jelenítsen meg egy szerkesztőablakot is, amely lehetővé teszi, hogy a felhasználó a csúszka használata helyett számszerűen írja be az értéket. Ha szerkesztőablakot alkalmaz erre a célra, aktualizálja annak értékét; máskülönben egy szövegzónában jelenítse meg a csúszka értékét. A következő ábrán csúszka és szerkesztőablak jellegzetes kombinációja látható:



Szerkesztőablakkal társított csúszka

Szöveg

Amikor a címkék nem elegendőek, használjon szövegzónákat a párbeszédpanel különálló zónáinak vagy területeinek azonosítására. A szövegzónákat állapot visszajelzésére vagy emlékeztetőként is használhatja, beleértve a hibaüzeneteket és a figyelmeztetéseket.

A szöveg legyen lényegretörő és félreérthetetlen. Olyan kifejezésekkel írja le az opciókat és az adatbeviteli zónákat, amelyeket a felhasználó is használna. Az Érvénytelen adat a listaablakban például kevés információt hordoz magában. A Fólia nem létezik üzenet sokkal több segítséget nyújt.

Az üzenetek kapcsolódjanak azokhoz a zónákhoz, amelyekre vonatkoznak.

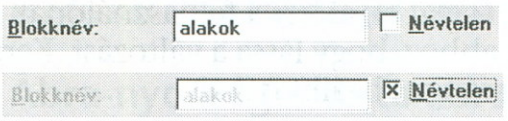
A zónák egy csoportját vagy a párbeszédpanel egy részét azonosító címkéket azon zónák *fölé* helyezze, amelyekre vonatkoznak.

Kapcsolók

Az azonos témához tartozó kapcsolók által vezérelt opciókat tegye egy csoportba.

Használjon olyan egyszerű kapcsolót, amely azt szabályozza, hogy más zónák, gyakran sorok vagy oszlopok, aktívak legyenek-e vagy sem. Ebben az esetben a kapcsolónak szembeötlőnek kell lennie. Ha a

kapcsoló csak egy zónát vezérel, akkor a kérdéses zónától jobbra is elhelyezheti. Erre példa a Névtelen kapcsoló a Blokk készítése párbeszédpanelben:



Zónát engedélyező vagy letiltó kapcsoló

Hibakezelés

A párbeszédpanelek a hibaüzeneteket és a figyelmeztetéseket egy szövegzóna, az úgynevezett *hibazóna* (`errtile`), vagy pedig beágyazott figyelmeztető ablak segítségével jeleníthetik meg. Az alábbi irányelvek mindkét módszerre vonatkoznak:

- A hibaüzenetek egész mondatok legyenek, nagybetűvel kezdve, és ponttal lezárva.
- A hibaüzenet magyarázza el a létező, illetve várható problémát. A fogalmazás legyen világos.
- Amennyiben lehetséges, a hiba jelzése után a párbeszédpanel fókuszát irányítsa arra a zónára, amely azt előidézte.

Hibazónák

A hibazónát kisebb hibákra utaló üzenetek és figyelmeztetések céljára használja, főleg olyanokéra, amelyek elírásból vagy egyéb téves adatbevitelből erednek.

Ne jelezze a hibákat a státuszüzenetek megjelenítésére használt zónákban. Ezek fölött könnyű átsiklani.

Amennyiben hibazónát alkalmaz, az a párbeszédpanel alsó részén jelenjen meg. Használja fel a szabvány `errtile` zónát, melynek leírását a "Párbeszédpanelek kilépő nyomógombjai és hibazóna" című résznél a 667. oldalon találhatja.

Figyelmeztető ablakok

Egy olyan szabvány figyelmeztető ablakot, amelyben egyetlen OK nyomógomb található, az AutoLISP illetve az ADS **`alert`** függvényével jeleníthet meg. A figyelmeztető ablakokat súlyos vagy potenciálisan végzetes hibák esetén alkalmazza, de kerülje túlzott használatukat. A figyelmeztető ablakok választ várnak a felhasználótól. Emiatt idegesítőek lehetnek, különösen ha csak kisebb hibát jeleznek, vagy éppen eltakarják a kijavításra szoruló adatot.

Használja a figyelmeztető ablakokat arra, hogy figyelmeztesse a felhasználót: az általa elindított művelet adatokat tehet tönkre, vagy

túlságosan időigényes. Az ilyen típusú figyelmeztető ablakok adjanak lehetőséget a felhasználónak a művelet folytatása vagy elvetése közötti választásra. Amennyiben a figyelmeztető ablak választási lehetőséget kínál (például Folytat vagy Mégsem), akkor azt Önnek kell megszerkesztenie.

Amennyiben a figyelmeztető ablak választási lehetőséget nyújt a felhasználónak, az ablak szövege először ismertesse a problémát, majd tegye fel a következő műveletre vonatkozó kérdést. Ezekben az esetekben különösen fontos, hogy a folytatást jelentő nyomógomb egy olyan igével legyen feliratozva, amely azt fejezi ki, ami történni fog. Ebben az összefüggésben a Felülír például sokkal egyértelműbb, mint az OK, és egyben segítség a gyakorlottabb felhasználóknak is, akik már sokadszor találkoznak ezzel a figyelmeztető ablakkal, és átsiklanak a szöveg fölött.

Hacsak nem következett be végzetes hiba, mindig adjon módot a felhasználónak, hogy visszatérjen az előző lépéshez vagy "megmeneküljön" a figyelmeztető ablakot előhívó művelettől.

A párbeszédpanel alapértelmezés szerinti nyomógombja rendszerint az OK, vagy annak megfelelője. Ha azonban a figyelmeztető ablakban ismertetett helyzet komoly következményekkel jár, az Mégsem nyomógombot vagy annak megfelelőjét tegye meg alapértelmezés szerintinek.

Kapcsolódó dokumentáció

A grafikus felhasználói felülettel kapcsolatos irányelveket és tervezési filozófiát nagyobb részletességgel tárgyalják a következő kiadványok:

- Apple Computer, Inc., *Apple Human Interface Guidelines: The Apple Desktop Interface*. Menlo Park: Addison-Wesley, 1987.
- International Business Machines Corp., *IBM System Application Architecture. Common User Access Guide to User Interface Design*. Boca Raton: IBM, 1991. (IBM publication number SC34-4289-00.)
- International Business Machines Corp., *IBM System Application Architecture. Common User Access Advanced Interface Design Reference*. Boca Raton: IBM, 1991. (IBM publication number SC34-4290-00.)

19

A PÁRBESZÉDVEZÉRLŐ NYELV

Ez a fejezet a Párbeszédvezérlő nyelv (Dialog Control Language, DCL) által kezelt valamennyi zónát, és az azokhoz tartozó attribútumokat tárgyalja. Mivel az attribútumok sok zónában közösek, ezért elsőnek ezekről esik szó. Ezután a DCL zónák következnek kategóriájuk és funkciójuk szerint. A fejezet végén az összes előre definiált DCL zónát tartalmazó katalógus található.

Attribútumok

A zóna attribútumai annak elrendezését és funkcióit határozzák meg. Egy attribútum hasonló a programozási nyelvekben használt változókhoz: névből és értékből áll. Az érték típusa az alábbiak egyike lehet:

- *Egész szám.* A távolságot --például egy zóna magasságát-- megadó számértékek (mind az egész, mind a valós számok) *karakterszélesség*- vagy *karaktermagasság*-egységekben kifejezett értékek.
- *Valós szám.* A tört alakú számoknak is számjeggyel kell kezdődniük: például `0.1`, *nem* pedig `.1`.
- *Idézőjeles karakterlánc.* A karakterlánc idézőjelek (`""`) közé zárt szöveget tartalmaz. Az attribútumértékek különbséget tesznek kis- és nagybetűk között: `B1` nem ugyanaz, mint `b1`. Ha a karakterláncban idézőjelet kell elhelyezni, akkor azt egy visszahajló törtjelnek (`\"`) kell megelőznie. Az idézőjelek közti karakterlánc tartalmazhat más vezérlő karaktereket is. A DCL által felismert karakterek az alábbi táblázatban láthatók.

A DCL karakterláncokban megengedett vezérlő karakterek

Vezérlő karakterek	*Jelentés*
`\"`	idézőjel (beágyazott)
`\\`	visszahajló törtjel
`\n`	újsor
`\t`	vízszintes tabulátor

- *Foglalt szó.* A foglalt szó olyan, alfanumerikus karakterekből álló azonosító, amely egy betűvel kezdődik. Sok attribútum igényel például egy `true` vagy `false` értéket. A foglalt szavak különbséget tesznek a kis- és nagybetűk között: `True` nem ugyanaz, mint `true`.

Megjegyzés A foglalt szavakhoz hasonlóan az attribútumok nevében sem egyenértékűek a kis- és nagybetűk: nem hivatkozhat a width attribútumra `Width` néven.

Az alkalmazói programok az attribútumokat mindig karakterláncként veszik át. Amennyiben az alkalmazás számértékeket használ, azokat előbb karakterlánccá kell konvertálnia, majd vissza kell alakítania őket számmá. További információ található a zóna értékek kezeléséről egy AutoLISP programon belül a "Zónák kezelése" című résznél, a 309. oldalon.

Egyes attribútumok -- például a `width` és a `height` -- minden zóna esetében megtalálhatók. Az attribútumok meghatározása nem kötelező. Sok attribútumnak van alapértelmezés szerinti értéke, amely

érvényre jut, ha az attribútum nincs meghatározva. Más attribútumok kifejezetten bizonyos zónához tartoznak—ilyen például egy ábra `color` (háttérszín) attribútuma. Az AutoCAD hibát jelezhet, ha ezt az attribútumot másfajta zónához próbálja hozzárendelni. Általában azonban nem veszi figyelembe az attribútumot.

Tiltott attribútumok

Ne használja a következő attribútumokat a DCL fájlokban.

`horizontal_margin`

`vertical_margin`

`type`

Előre definiált attribútumok

Ebben a részben a PDB szolgáltatás által definiált attribútumokat ismertetjük. Az alábbi táblázat betűrendbe foglalja az előre definiált attribútumokat. Az attribútumokat részletesebben a táblázat után ismertetjük.

Az előre definiált attribútumok összefoglalása

Attribútum neve	*Hozzá tartozó zóna*	*Jelentése (ha meg van adva vagy true)*
`action`	minden aktív zóna	AutoLISP akciókifejezés
`alignment`	minden zóna	vízszintes vagy függőleges pozíció egy csoportban
`allow_accept`	szerkesztőablak, ábra-nyomógomb és listaablak	kiválasztásakor aktivizálja az `is_default` nyomógombot
`aspect_ratio`	ábra, ábra-nyomógomb	az ábra függőleges és vízszintes méretének az aránya
`big_increment`	csúszka	egy lépéshez tartozó növekmény (a platformnak támogatnia kell a növekményes vezérlő eszközöket)
`children_alignment`	sor, oszlop, rádiósor, rádióoszlop, bekeretezett sor, bekeretezett oszlop, bekeretezett rádiósor és bekeretezett rádióoszlop	a csoport gyermekeinek igazítása

Az előre definiált attribútumok összefoglalása (*folytatás*)

Attribútum neve	*Hozzá tartozó zóna*	*Jelentése (ha meg van adva vagy true)*
`children_fixed_height`	sor, oszlop, rádiósor, rádióoszlop, bekeretezett sor, bekeretezett oszlop, bekeretezett rádiósor és bekeretezett rádióoszlop	a csoport gyermekeinek magassága nem növekedik elrendezés közben
`children_fixed_width`	sor, oszlop, rádiósor, rádióoszlop, bekeretezett sor, bekeretezett oszlop, bekeretezett rádiósor és bekeretezett rádióoszlop	a csoport gyermekeinek szélessége nem növekedik elrendezés közben
`color`	ábra, ábra-nyomógomb	az ábra háttér- (kitöltési-) színe
`edit_limit`	szerkesztőablak	a felhasználó által beírható karakterek maximális száma
`edit_width`	szerkesztőablak, legördülő lista	a zóna szerkesztési (adatbeviteli) mezőjének szélessége
`fixed_height`	minden zóna	a magasság nem növekedhet elrendezés közben
`fixed_width`	minden zóna	a szélesség nem növekedik elrendezés közben
`height`	minden zóna	a zóna magassága
`initial_focus`	párbeszéd	a kezdeti fókusszal kitüntetett zóna key attribútuma
`is_bold`	szöveg	kövér (bold) megjelenítés (a platformnak támogatnia kell a kövér szöveget)
`is_cancel`	nyomógomb	ez a nyomógomb aktivizálódik a mégsem billentyű—általában Ctrl+C vagy Esc—lenyomásakor (a platformnak támogatnia kell a billentyűzet-rövidítéseket)
`is_default`	nyomógomb	ez a nyomógomb aktivizálódik az elfogad billentyű—általában ↵—lenyomásakor (a platformnak támogatnia kell a billentyűzet-rövidítéseket)

Az előre definiált attribútumok összefoglalása (*folytatás*)

Attribútum neve	*Hozzá tartozó zóna*	*Jelentése (ha meg van adva vagy true)*
`is_enabled`	minden aktív zóna	kezdetben a zóna engedélyezett
`is_tab_stop`	minden aktív zóna	a zóna tabulátorhelynek számít (a platformnak támogatnia kell a billentyűzet-rövidítéseket)
`key`	minden aktív zóna	a zóna alkalmazás által használt neve
`label`	bekeretezett sor, bekeretezett oszlop, bekeretezett rádiósor, bekeretezett rádióoszlop, nyomógomb, párbeszéd, szerkesztőablak, listaablak, legördülő lista, rádiógomb, szöveg és kapcsoló	a zóna képernyőn megjelenő címkéje
`layout`	csúszka	a csúszka vízszintesen vagy függőlegesen helyezkedik-e el
`list`	listaablak, legördülő lista	a listában megjeleníteni kívánt kezdeti értékek
`max_value`	csúszka	a csúszka maximális értéke
`min_value`	csúszka	a csúszka minimális értéke
`mnemonic`	minden aktív zóna	a zónához tartozó mnemonik karakter (a platformnak támogatnia kell a billentyűzet-rövidítéseket)
`multiple_select`	listaablak	a listaablak megengedi több tétel kiválasztását
`small_increment`	csúszka	egy lépéshez tartozó növekmény (a platformnak támogatnia kell a növekményes vezérlő eszközöket)
`tabs`	listaablak, legördülő lista	tabulátorhelyek a lista megjelenítéséhez
`value`	szöveg, aktív zónák (nyomógombok és ábra-nyomógombok kivételével)	a zóna kezdeti értéke
`width`	minden zóna	a zóna szélessége

Az alábbi katalógus az előre definiált DCL zónák által használt attribútumokat tartalmazza.

action

action = "(*függvény*)";

Egy AutoLISP kifejezést ad meg, amely a zóna kiválasztásakor valamilyen művelet végrehajtására szolgál. *Visszajelzés* néven is ismeretes. A zónák néhány fajtájánál akkor is bekövetkezhet akció, amikor a felhasználó a fókuszt egy másik zónára irányítja át.

Lehetséges értéke egy érvényes AutoLISP kifejezést tartalmazó, idézőjelek közötti karakterlánc. Az ADS rendszerben nem adhatja meg az `action` attribútumot, mindkét nyelven írhat azonban olyan alkalmazást, amelyek támogatják a párbeszépanel kezelését. Egy zónához csak egy akció tartozhat. Ha az alkalmazás hozzárendel egy akciót (az **`action_tile`** függénnyel), az felülírja az `action` attribútumot.

alignment

alignment = *pozíció*;

A zónának a csoporton belüli vízszintes és függőleges eligazítását (egyenletes elrendezését) határozza meg .

Egy oszlopon belüli zóna lehetséges értékei `left`, `right` vagy `centered` (alapértelmezés: `left`).

Egy soron belüli zóna lehetséges értékei `top`, `bottom vagy centered` (alapértelmezés: `centered`).

A csoport hossztengelye mentén nem adhatja meg az eligazítást. A csoportban az első és utolsó zóna mindig az oszlop, illetve sor végéhez igazodik. A csoportban a többi zóna egyenletesen oszlik el, hacsak az elosztást nem változtatja meg helykitöltő beillesztési pontok használatával (További információ a "helykitöltő (spacer_0)" című résznél, a 683. oldalon található).

allow_accept

allow_accept = *true-false*;

Megadja, hogy a zóna aktivizálódjon-e, ha a felhasználó megnyomta az "elfogad" billentyűt (általában ↵). Ha értéke `true`, és a felhasználó megnyomja az "elfogad" billentyűt, az annak felel meg, mintha az alapértelmezés szerinti nyomógombot (ha van ilyen) "nyomná meg." Egyes platformokon a dupla kattintás a zónában az "elfogad" megnyomásával egyenértékű. Az alapértelmezés szerinti nyomógomb az a `button` zóna, amelynek `is_default` attribútuma `true` értékű. Az attribútum alapértelmezés szerinti értéke `false`.

aspect_ratio

aspect_ratio = *valós szám;*

Az ábra szélességének és magasságának arányát határozza meg (szélesség osztva a magassággal). Amennyiben nulla (0.0), a zóna az ábra méretéhez igazodik.

Lehetséges értékei lebegőpontos számok (alapértelmezés: nincs).

big_increment

big_increment = *egész szám;*

A csúszka az adott növekménnyel történő léptetéséhez felhasznált értékeket határozza meg. A big_increment alapértelmezés szerinti értéke a teljes tartomány egytized része. Az értéknek belül kell maradnia a min_value és a max_value által meghatározott tartományon.

Megjegyzés Nem minden platform biztosít kezelőszerveket a csúszka meghatározott növekménnyel történő léptetésére. Amelyek mégis, azok általában nyilakat (például a görgetősáv végein) vagy (+) és (++) jeleket alkalmaznak, ahol a megkettőzött plusz jel a nagyobb növekményt jelzi. Ha a platform nem biztosít ilyen kezelőszerveket, az attribútumot a program figyelmen kívül hagyja.

children_alignment

children_alignment = *pozíció;*

A csoporthoz tartozó összes zóna alapértelmezés szerinti eligazítását adja meg (az alignment attribútumhoz hasonlóan). *Nem* írja felül egy gyermek igazítási attribútumát, ha az külön meg van adva.

Az oszlopok esetében a lehetséges értékek: left, right vagy centered (alapértelmezés: left).

A sorok esetében lehetséges értékek: top, bottom vagy centered (alapértelmezés: centered).

children_fixed_height

children_fixed_height = *true-false;*

A csoportban található összes zóna alapértelmezés szerinti magasságát adja meg (a `height` attribútumhoz hasonlóan). *Nem* írja felül egy gyermek height attribútumát, ha az külön meg van adva.

Lehetséges értékek `true` vagy `false` (alapértelemzés: `false`).

Megjegyzés Körültekintően használja a `fixed_` attribútumokat. Az alapértelemzések következetlen felülírása következetlen elrendezést eredményez.

children_fixed_width

children_fixed_width = *true-false;*

A csoportban található összes zóna alapértelmezés szerinti szélességét adja meg (a `width` attribútumhoz hasonlóan). *Nem* írja felül egy gyermek width attribútumát, ha az külön meg van adva.

Lehetséges értékek `true` vagy `false` (alapértelmezés: `false`).

Megjegyzés Körültekintően használja a `fixed_` attribútumokat. Az alapértelemzések következetlen felülírása következetlen elrendezést eredményez.

color

color = *színnév;*

Megadja egy ábra háttér- (kitöltési) színét. Lehetséges értékei egész számok vagy foglalt szavak (alapértelmezés: `7`), amelyek AutoCAD színszámként vagy az alábbi táblázatban található szimbolikus nevek egyikeként vannak megadva.

A color attribútumban használható szimbolikus nevek

Szimbolikus név	*Jelentés*
`dialog_line`	Aktuális párbeszédpanel-vonal színe
`dialog_foreground`	Aktuális párbeszédpanel-előtér színe (szöveg esetén)
`dialog_background`	aktuális párbeszédpanel-háttér színe
`graphics_background`	Az AutoCAD grafikus képernyőjének aktuális háttere (általában 0)

A color attribútumban használható szimbolikus nevek (*folytatás*)

Szimbolikus név	*Jelentés*
`black`	AutoCAD szín = 0 (fekete) (fekete háttér esetén világos színben jelenik meg)
`red`	AutoCAD szín = 1 (vörös)
`yellow`	AutoCAD szín = 2 (sárga)
`green`	AutoCAD szín = 3 (zöld)
`cyan`	AutoCAD szín = 4 (világoskék)
`blue`	AutoCAD szín = 5 (kék)
`magenta`	AutoCAD szín = 6 (bíbor)
`white` `graphics_foreground`	AutoCAD szín = 7 (fehér) (világos háttér esetén fekete színben jelenik meg)

A `graphics_background` és `graphics_foreground` nevek a `black` és `white` alternatívájaként szolgálnak. A különleges színek használata zavaró lehet, mivel az általuk megjelenített szín az aktuális AutoCAD konfiguráció függvényében változik. Egy ábrában megjelenített diakép vektorai gyakran fekete vagy fehér színűek. Amennyiben az Ön ábrazónája az első megjelenítés alkalmával üresnek látszik, próbálja meg annak `color` attribútumát `graphics_background` vagy `graphics_foreground` értékre változtani.

edit_limit

`edit_limit = egész szám;`

Megadja a felhasználó által a szerkesztőablakba beírható maximális karakterszámot. Lehetséges értéke egy egész szám (alapértelmezés: `132`). Amikor a felhasználó eléri ezt a határt, a megjelenítő eszköz meghajtója nem fogad el további karaktereket (a Backspace és a Del kivételével). A szerkesztési határ megengedett értéke 256 karakter lehet.

edit_width

`edit_width = szám;`

A szerkesztőablak szerkesztési (adatbeviteli) részének karakterszélesség-egységekben kifejezett szélességét adja meg—ez az `edit_box` zóna ténylegesen bekeretezett része. Az értékek lehetnek egész és valós számok is. Ha az `edit_width` nincs megadva vagy nulla, és a zóna

szélessége nem rögzített, akkor a mező kitölti a rendelkezésre álló helyet. Ha az `edit_width` nullától különböző érték, akkor a mező a zóna által elfoglalt helyen belül jobbra igazítva helyezkedik el. Ha elrendezési okokból a zónát "tágítani" szükséges, akkor a PDB szolgáltatás üres helyet illeszt a címke és az adatbeviteli mező közé.

fixed_height

`fixed_height = true-false;`

Megadja, hogy a zóna magassága kitöltheti-e a rendelkezésre álló helyet. Amennyiben az attribútum értéke `true`, a zóna nem tölti ki az elrendezés/eligazítás során keletkezett üres területet.

Lehetséges értékei `true` vagy `false` (alapértelmezés: `false`).

fixed_width

`fixed_width = true-false;`

Megadja, hogy a zóna szélessége kitöltheti-e a rendelkezésre álló helyet. Amennyiben az attribútum értéke `true`, a zóna nem tölti ki az elrendezés/eligazítás során keletkezett üres területet.

Lehetséges értékei `true` vagy `false` (alapértelmezés: `false`).

height

`height = szám;`

Megadja a zóna magasságát. Lehetséges értéke egy egész vagy egy valós szám, amely karaktermagasság-egységekben fejezi ki a távolságot. Csak akkor adja meg ezt az értéket, ha az alapértelmezés szerinti érték nem biztosít megfelelő megjelenést. Az ábrazónák és ábra-nyomógombok esetében azonban meg kell adni a magasságot.

A `height` attribútum a zóna *minimális* magasságát adja meg. Ez a méret növekedhet a zóna elrendezésekor, hacsak a magasság a `fixed_` attribútumok egyikével nincs rögzítve. Az alapértelmezés szerinti értékeket a rendszer dinamikusan rendeli hozzá úgy, ahogyan azt az elrendezés megköveteli.

A *karaktermagasság egysége* a definíció szerint a képernyőn megjelenített karakterek maximális nagysága (beleértve a sorok közötti hézagot is). Az egységek értéke a platformtól függ.

initial_focus

```
initial_focus = "karakterlánc";
```

Annak zónának a `key` attribútumát adja meg a párbeszépanelben, amelyre a kezdeti billentyűzetfókusz irányul. Lehetséges értéke egy idézőjeles karakterlánc (nincs alapértelmezés szerinti értéke).

is_bold

```
is_bold = true-false;
```

Megadja, hogy a szöveg kövér (bold) karakterekkel jelenjen-e meg. Lehetséges értékei `true` vagy `false` (alapértelmezés: `false`). Ha értéke `true`, a szöveg kövér karakterekkel jelenik meg. (Nem minden platform támogatja.)

is_cancel

```
is_cancel = true-false;
```

Megadja, hogy kiválasztja-e a felhasználó a nyomógombot, amikor megnyomja az mégsem billentyűt (Esc vagy Ctrl+C). Értéke `true` vagy `false` lehet (alapértelmezés: `false`).

Amennyiben az akciókifejezés nyomógombjának `is_cancel` attribútuma `true` értékre van állítva, és nem lép ki a párbeszédpanelből (nem hívja meg a **`done_dialog`** függvényt), a párbeszédpanel automatikusan lezárul az akciókifejezés végrehajtása után, a DIASTAT rendszerváltozó értéke pedig 0 lesz.

A párbeszédpanelnek csak egy nyomógombjához tartozhat `true` értékű `is_cancel` attribútum.

is_default

```
is_default = true-false;
```

Megadja, hogy az alapértelmezés szerinti nyomógomb kerül-e kiválasztásra, amikor a felhasználó megnyomja az "elfogad" billentyűt (↵ a legtöbb olyan platform esetében, amely támogatja a gyorsító billentyűket). Értéke `true` vagy `false` lehet (alapértelmezés: `false`). Ha a felhasználó egy olyan `edit_box`, `list_box` vagy `image_button` zónában van, amelynek `allow_accept` attribútuma `true` értékű, akkor az "elfogad" billentyű lenyomásával vagy (listaablak és ábranyomógomb esetében) dupla kattintással szintén az alapértelmezés szerinti nyomógombot választja ki. *Nem* az alapértelmezés szerinti nyomógombot választja ki, ha az "elfogad" billentyű lenyomásakor egy másik nyomógombon van a fókusz. Ebben az esetben az a nyomógomb kerül kiválaszrásra, amelyiken a fókusz van.

A párbeszédpanelnek csak egy nyomógombjához tartozhat `true` értékű `is_default` attribútum.

is_enabled

```
is_enabled = true-false;
```

Megadja, hogy a zóna kezdetben halvány (nem elérhető) legyen-e. Értéke `true` vagy `false` lehet (alapértelmezés: `true`). Ha értéke `false`, a zóna kezdetben halvány lesz.

is_tab_stop

```
is_tab_stop = true-false;
```

Megadja, hogy a zóna fogadja-e a billentyűzetfókuszt, amikor a felhasználó a Tab billentyű segítségével közlekedik a zónák között. Értéke `true` vagy `false` lehet (alapértelmezés: `true`). Ha a zóna le van tiltva, akkor a tabulátorral abban az esetben sem lehet rálépni, ha az attribútum értéke `true`. Amennyiben értéke `false`, a zóna nem tabulátorhely.

key

```
key = "karakterlánc";
```

Egy ASCII nevet ad meg, amelyet a program az adott zónára történő hivatkozásra használ. Lehetséges értéke egy idézőjeles karakterlánc (alapértelmezés szerinti értéke nincs). Egy adott párbeszédpanelen belül minden `key` értéknek egyedinek kell lennie. Ez a karakterlánc különbséget tesz a kis- és nagybetűk között: ha a key attribútumnak `NagyZóna` értéket ad meg, nem hivatkozhat rá a `nagyzóna` értékkel.

Mivel a `key` értéke a felhasználó számára láthatatlan, bármilyen nevet választhat számára (amíg az egyedi a párbeszédpanelben), ezért a `key` attribútumokat a több nyelven forgalmazott alkalmazások esetében sem szükséges lefordítani.

label

```
label = "karakterlánc";
```

A zónában megjelenítendő szöveget adja meg. Lehetséges értéke egy idézőjeles karakterlánc (alapértelmezés: egy üres karakterlánc, `" "`). A `label` (címke) attribútum szövegének elhelyezkedése zóna-specifikus.

A címke attribútum megadhat egy mnemonik karaktert a zóna számára. A legtöbb platform esetében a mnemonikot aláhúzással jelölik a címke attribútumban.

Ha a címke karakterláncában egy karaktert az *és* jel(&) előz meg, akkor az a karakter lesz a mnemonik. A karakternek nem kell egyedinek lennie a párbeszédpanelben: ha több zónának ugyanaz a mnemonikja, a felhasználó az adott billentyűvel lépkedhet az azonos mnemonikkal rendelkező zónák közöttt.

Megjegyzés Nem minden felhasználói grafikus felület támogatja a mnemonikokat.

A mnemonikok megváltoztatják a fókuszt; vagyis sohasem választanak ki egy zónát. Ha a felhasználó egy olyan zóna számára ad meg egy mnemonikot, amely tételek csoportját tartalmazza -- például egy csoport vagy egy listaablak --, akkor a fókusz a zóna első olyan tételére kerül, amely tabulátorhely. Bármelyik aktív zóna tabulátorhelynek számít, amíg az `is_tab_stop` attribútumának értéke nem `false`.

Megjegyzés A `mnemonic` attribútum szintén mnemonik karaktert ad meg.

layout

```
layout = pozíció;
```

A csúszka elhelyezkedését adja meg. Értéke `horizontal` vagy `vertical` lehet (alapértelmezés: `horizontal`). A vízszintes csúszkák esetében az értékek balról jobbra, függőleges csúszkák esetében alulról felfelé növekszenek.

list

```
list = "karakterlánc";
```

Megadja a `popup_list` vagy `list_box` zónában elhelyezett sorok kezdeti halmazát. Értéke egy idézőjeles karakterlánc lehet (nincs alapértelmezés szerinti értéke). A sorokat az újsor szimbólum (`\n`) választja el. A tabulátor karakterek (`\t`) minden sorban megjelenhetnek.

max_value

```
max_value = egész szám;
```

A csúszka által szolgáltatott értékek felső tartományát adja meg. Alapértelmezés szerinti maximális értéke `10000`. Ennek az értéknek előjelesnek, 16 bites egésznek valamint `32768`-nál kisebbnek kell lennie.

min_value

```
min_value = egész szám;
```

A csúszka által szolgáltatott értékek alsó tartományát adja meg. Alapértelmezés szerinti minimális értéke `0`. Ennek az értéknek előjelesnek, 16 bites egésznek valamint `-32768`-nál nagyobbnak kell lennie.

A `min_value` értéke nagyobb is lehet, mint a `max_value`. Egyes platformok esetében ez megfordítja a sorrendet, amelyben az értékek megjelennek a képernyőn, és amelyben a csúszka mozgatáskor az értékeket visszaadja.

mnemonic

```
mnemonic = "karakter";
```

Megadja a zónához tartozó mnemonik billentyűt. A legtöbb platform esetében a mnemonik a zóna címkéjében aláhúzással jelölt karakter. Értéke egy idézőjeles karakterlánc vagy *egyetlen* karakter lehet (nincs alapértelmezés szerinti értéke). A karakternek a zóna `label` attribútumában szereplő egyik betűnek kell lennie. A legtöbb platform esetében nem szükséges, hogy a karakterlánc egyedi legyen a párbeszédpanelben: ha több zónához tartozik ugyanaz a karakter, akkor a megfelelő billentyű lenyomásával a zónák egymás után kerülnek sorra.

A felhasználó szempontjából a mnemonikok esetében nincs különbség a kis- és nagybetűk használata között: például, ha egy nyomógomb mnemonikja az *A*, akkor mind az **a**, mind az **A** lenyomása az adott nyomógombra viszi a fókuszt. A DCL fájlban azonban a mnemoniknak a zóna `label` attribútumában szereplő karakterek egyikének kell lennie és pontosan úgy kell megadni, ahogyan a `label` karakterláncban szerepel.

Megjegyzés Nem minden grafikus felhasználói felület támogatja a mnemonikok használatát.

A mnemonikok megváltoztatják a fókuszt; vagyis sohasem választanak ki egy zónát. Ha a felhasználó egy olyan zóna számára ad meg egy mnemonikot, amely tételek csoportját tartalmazza -- például egy csoport vagy egy listaablak --, akkor a fókusz a zóna első olyan tételére kerül, amely tabulátorhely. Bármelyik aktív zóna tabulátorhelynek számít, amíg az `is_tab_stop` attribútumának értéke nem `false`.

Megjegyzés A `label` attribútum is megadhat egy mnemonik karaktert.

multiple_select

multiple_select = *true-false;*

Megadja, hogy egy `list_box` zónában lehetséges-e több tétel kiválasztása és kiemelt megjelenítése. Értéke lehet `true` vagy `false` (alapértelmezés: `false`). Ha értéke `true`, egyszerre több tételt is ki lehet választani. A több tétel kiválasztásának módszere a platformtól függ.

small_increment

small_increment = *egész szám;*

A csúszka adott növekménnyel történő léptetéséhez felhasznált értékeket határozza meg. A `small_increment` alapértelmezés szerinti értéke a teljes tartomány egytized része. Az értéknek belül kell maradnia a `min_value` és a `max_value` által meghatározott tartományon. Ez az attribútum opcionális.

Megjegyzés Nem minden platform biztosít kezelőszerveket a csúszka meghatározott növekménnyel történő léptetésére. Amelyek mégis, azok általában nyilakat (például a görgetősáv végein) vagy (+) és (++) jeleket alkalmaznak, ahol a megkettőzött plusz jel a nagyobb növekményt jelzi. Ha a platform nem biztosít ilyen kezelőszerveket, az attribútumot nem veszik figyelembe.

tabs

tabs = *"karakterlánc";*

Megadja a tabulátorhelyek elhelyezkedését karakterszélesség-egységekben. Értéke egy szóközzel elválasztott, egész vagy lebegőpontos számokat magában foglaló idézőjeles karakterlánc lehet (alapértelmezés szerinti értéke nincs). Ezek az értékek a szöveg oszlopainak függőleges eligazítására szolgálnak a `popup_list` vagy a `list_box` zónában (Példaképpen tekintse meg az AutoCAD Fóliák beállítása párbeszédpanelt, amely `acad_mlayer` néven az *acad.dcl* fájlban található.)

Akövetkező programrészlet például 8 karakterenként ad meg egy tabulátorhelyet.

```
tabs = "8 16 24 32";
```

Megjegyzés Amennyiben egy legördülő listában tabulátorhelyeket használ a szöveg oszlopainak eligazítására, annak megjelenése platformról platformra változhat.

value

```
value = "karakterlánc";
```

Megadja egy zóna kezdeti értékét. Értéke egy idézőjeles karakterlánc lehet (alapértelmezés: egy üres karakterlánc, `" "`). A zóna értékének jelentése függ a zóna fajtájától. A zóna értéke futásidőben változhat a felhasználó által bevitt adatok vagy **`set_tile`** hívások következtében.

A párbeszédpanel elrendezését a zóna `value` (érték) attribútuma nem befolyásolja. Miután az elrendezés befejeződött és a párbeszédpanel megjelent, a **`new_dialog`** az érték attribútum felhasználásával állítja be a párbeszédpanel minden zónájának kezdeti értékét. A zóna érték attribútumának nincs hatása a zónák méretére és kiosztására a párbeszédpanelben.

width

```
width = szám;
```

Megadja a zóna szélességét. Értéke egy egész vagy valós szám lehet, amely karkterszélesség-egységekben fejezi ki a távolságot. Csak akkor adja meg ezt az értéket, ha az alapértelmezés szerinti érték nem biztosít megfelelő megjelenést. Az ábrazónák és ábra-nyomógombok esetében azonban meg kell adni a szélességet.

A zóna `width` attribútuma csak a *minimális* szélességet adja meg. Ez a méret növekedhet a zóna elrendezésekor, hacsak a szélesség nincs a `fixed_` attribútumok egyikével rögzítve. Az alapértelmezés szerinti értékek hozzárendelését a rendszer dinamikusan végzi el úgy, ahogyan azt az elrendezés megköveteli.

A *karakterszélesség egysége* a definíció szerint az az érték, amely az összes kis- és nagybetű szélességének átlagolásával, illetve a képernyő szélességének 80-nal történő osztásával kapott értékek közül a kisebb (az átlagos szélesség `(width(A...Z) + width(a...z)) / 52`). Ezen egység értéke a platformtól függ.

Felhasználó által definiált attribútumok

A zónák definiálásakor megadhat saját attribútumokat is. Az attribútum neve bármilyen érvényes név lehet, amely nem ütközik az előző részben leírt és "Az előre definiált attribútumok összefoglalása" című, 651. oldalon található táblázatban összefoglalt, előre definiált attribútumokkal. Az attribútum neve -- hasonlóan a kulcsszavakhoz-- betűket, számokat és aláhúzás (_) karaktert tartalmazhat, továbbá betűvel kell kezdődnie.

Megjegyzés Amennyiben a felhasználó által definiált attribútum ütközik egy előre definált attribútummal, a PDB szolgáltatás nem ismeri fel, hogy az új attribútum, és az Ön által megadott értéket az

előre definiált attribútummal próbálja használni. Az ilyen hibát nagyon nehéz felderíteni.

Az attribútumhoz rendelt értéket és annak jelentését az adott alkalmazás határozza meg. A felhasználó által definiált attribútumok típusának meg kell felelnie az "Attribútumok" című résznél, a 650. oldalon ismertetett típusoknak.

Az attribútumok definiálása az alkalmazás-specifikus, úgynevezett kliensadatok definiálásához hasonlítható. Mindkét módszer lehetővé teszi, hogy a PDB szolgáltatás az Ön által megadott adatokat kezelje. A felhasználó által definiált adatok csak olvashatók; vagyis statikusak (állandóak) mindaddig, amíg a párbeszédpanel aktív. Kliensadatot akkor kell használnia, ha az értékek dinamikus, futásidőben történő megváltoztatására szüksége van. A végfelhasználók a felhasználó által definiált attribútumok értékét az alkalmazás DCL fájljában megszemlélhetik, a kliensadat viszont láthatatlan számukra.

Az AutoCAD Rajzi segédeszközök párbeszédpaneljének definíciója tartalmazza az `errmsg` nevű attribútum definícióját, amely egy egyedi karakterlánc-érték minden zóna esetében. Az `errmsg` értékét egy általános hibakezelő használja, amikor figyelmeztetést jelenít meg. A zóna például az alábbi értéket adhatja az `errmsg` attribútumnak:

```
errmsg = "a Háló Y kiosztása";
```

Amennyiben a felhasználó nem megfelelő értéket -- például negatív számot -- adott meg, az AutoCAD az alábbi hibaüzenetet jeleníti meg:

Érvénytelen a Háló Y kiosztása.

Az Érvénytelen szót és a pontot(.) a hibakezelő teszi az üzenethez.

A felhasználó által definiált attribútumok használhatók egy zóna értékének megengedett határok között tartására, vagy egy zóna által aktivizált beágyazott párbeszédpanel megnevezésére is (további információ a "Párbeszédpanelek egymásba ágyazása" című résznél, a 313. oldalon található).

A DCL zónák funkciók szerinti áttekintése

Ebben a részben a DCL zónák működés szerinti csoportosításban találhatók.

Előre definiált aktív zónák

Az AutoCAD PDB szolgáltatása rendelkezik beépített (előre definiált) zónákkal, melyeket önmagukban vagy összetettebb zónák alapjaiként lehet használni. Definíciójuk a *base.dcl* fájlban megjegyzésként szerepel (további információ a "A base.dcl és az acad.dcl fájl" című résznél, a 621. oldalon található.)

Amikor a felhasználó kiválaszt egy aktív zónát—például egy nyomógombot—, a párbeszédpanel közli ezt az őt kezelő alkalmazással. Bármelyik előre definiált aktív zónához tartozhat akció. Az akció hatása lehet látható a felhasználó számára, de lehet teljességgel belső lefolyású is (például egy státusz frissítése). Az akcióhoz egy indítékkód tartozik, amely jelzi, hogy mi váltotta azt ki. Az indíték jelentése az akciót kiváltó zóna típusától függ. Íme a választható, aktív zónák:

```
button                  popup_list
edit_box                radio_button
image_button            slider
list_box                toggle
```

Zónacsoportok

A zónákból összetett sorok vagy oszlopok (ezek gyűjtőneve *csoportok* vagy *részösszeállítások)* állíthatók össze. Elrendezési célból a sorokat vagy oszlopokat egyetlen zónaként kezeli a rendszer. A sor vagy oszlop lehet bekeretezett, és ellátható opcionális címkével is (keret nélküli csoportnak nem lehet címkéje).

A felhasználó csoportot nem, csak a csoporton belüli egyedülálló (kiválasztható, aktív) zónákat választhat ki. A csoportokhoz nem lehet akciót rendelni, kivéve a rádiósorokat és rádióoszlopokat. A következő zónák definiálnak csoportokat :

```
boxed_column            dialog
boxed_radio_column      radio_column
boxed_radio_row         radio_row
boxed_row               row
column
```

Dekoratív és tájékoztató jellegű zónák

Az itt felsorolt zónák *nem* hajthatnak végre akciókat és *nem* választhatók ki. Információ megjelenítésére, vizuális hangsúlyok kifejezésére használhatók, illetve a párbeszédpanel elrendezéséhez nyújtanak segítséget.

```
image                    spacer_0

text                     spacer_1

spacer
```

Szövegcsoportok

Egy szövegzónát -- mint bármely más zónát -- szegélyező tér vesz körül. Ez problémát jelenthet, ha Ön szövegdarabokat kíván összeilleszteni. Tegyük fel például, hogy a következő üzenetet kívánja megjeleníteni:

A pontos idő 0800 óra 37 perc.

Az aktuális értékeket (0800 és 37) a program biztosítja. A megjelenítéshez hozzon létre a `text_part` zónákból egy összeláncolt szövegből álló sort. A szövegrészeket függőlegesen is használhatja olyan bekezdések létrehozására, amelyek sorai között nincs túl sok hely.

A következő szövegcsoport-zónák a *base.dcl* fájlban definiált prototípusok:

```
concatenation            text_part

paragraph
```

Párbeszédpanelek kilépő nyomógombjai és hibazóna

A *base.dcl* fájl előre definiált nyomógomb-részösszeállításokat biztosít a párbeszédpanelből való kilépésre (vagy "visszavonulásra"). Használja ezeket a változatokat arra, hogy a megjelenés következetes maradjon az összes alkalmazásban. A *base.dcl* fájl az adott platform szabványainak figyelembevételével minden platformváltozatra definiál nyomógombokat, így ezen változatok használatával a platformmal való kompatibilitás is biztosítható.

"A kilépő nyomógomb szövegének megváltoztatása" című résznél, a 632. oldalon leírt `retirement_button` prototípus használatával a nyomógombok szövegét saját igényeihez igazíthatja.

`errtile`	`ok_cancel_help`
`ok_only`	`ok_cancel_help_errtile`
`ok_cancel`	`ok_cancel_help_info`

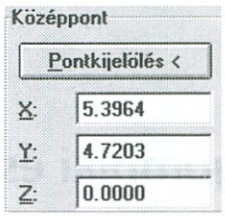
Tiltott zónák

Az Ön DCL fájljai nem használhatják a `cluster` és `tile` zónákat. Ne használja az alapvető kilépő nyomógomb-típusokat (`cancel_button`, `help_button`, `info_button` és `ok_button`) sem, hacsak nem definiálja át a "Párbeszédpanelek kilépő nyomógombjai és hibazóna" című résznél, a 667. oldalon leírt előre definiált kilépő nyomógomb-részösszeállításokat .

DCL zónák katalógusa

Ez a rész bemutatja az összes előre definiált DCL zónát. A zóna nevét követő szintaktikai utasítások a zónával kapcsolatos összes attribútumot tartalmazzák. Az egyes specifikus attribútumok működését a zóna leírása után ismertetjük.

bekeretezett oszlop




```
: boxed_column {
  alignment  children_alignment
  children_fixed_height  children_fixed_width
  fixed_height  fixed_width  height  label  width
}
```

A bekeretezett oszlopot keretvonal határolja. A párbeszédpanel elrendezése is bekeretezett oszlophoz hasonló. Amennyiben a bekeretezett oszlopnak címkéje is van, az a keret fölött, vagy abba beillesztve helyezkedik el, a grafikus felhasználói felülettől függően. Ha a címke hiányzik, vagy csak egy szóköz (`" "`), illetve üres (`""`) értékű, csak a keret jelenik meg.

`label` Címként jelenik meg. Helykihasználás szempontjából különbség lehet egy szóköz értékű, és egy üres címke között. (További információ az "Üres hely egy bekeretezett sor vagy oszlop körül" című résznél, a 631. oldalon található.)

bekeretezett rádióoszlop

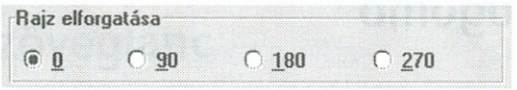
```
: boxed_radio_column {
  alignment  children_alignment
  children_fixed_height  children_fixed_width
  fixed_height  fixed_width  height  label  width
}
```

A bekeretezett rádióoszlopot keretvonal határolja. A címkét a bekeretezett oszlopnál tárgyalt módon kell kezelni.

label Címként jelenik meg. Ha a címke hiányzik, vagy csak egy szóköz (ez az alapértelmezés) , illetve üres (`""`) értékű, csak a keret jelenik meg. Helykihasználás szempontjából különbség lehet egy szóköz értékű, és egy üres címke között. (További információ az "Üres hely egy bekeretezett sor vagy oszlop körül" című résznél, a 631. oldalon található.)

value Az éppen kiválasztott rádiógomb `key` attribútumát adja meg (az `"1"` értékűt).

bekeretezett rádiósor




```
: boxed_radio_row {
  alignment  children_alignment
  children_fixed_height  children_fixed_width
  fixed_height  fixed_width  height  label  width
}
```

A bekeretezett rádiósort keret határolja. A címkét a bekeretezett sornál tárgyalt módon kell kezelni.

label Címként jelenik meg. Ha a címke hiányzik, vagy csak egy szóköz (ez az alapértelmezés) , illetve üres (`""`) értékű, csak a keret jelenik meg. Helykihasználás szempontjából különbség lehet egy szóköz értékű, és egy üres címke között. (További információ az "Üres hely egy bekeretezett sor vagy oszlop körül" című résznél, a 631. oldalon található.)

value Az éppen kiválasztott rádiógomb `key` attribútumát adja meg (az `"1"` értékűt).

bekeretezett sor

```
: boxed_row {
  alignment  children_alignment
  children_fixed_height  children_fixed_width
  fixed_height  fixed_width  height  label  width
}
```

A bekeretezett sort keretvonal határolja. Amennyiben a bekeretezett oszlopnak címkéje is van, az a keret fölött, vagy abba beillesztve helyezkedik el, a grafikus felhasználói felülettől függően.

`label` Címként jelenik meg. Ha a címke hiányzik, vagy csak egy szóköz (ez az alapértelmezés), illetve üres (`""`) értékű, csak a keret jelenik meg. Helykihasználás szempontjából különbség lehet egy szóköz értékű, és egy üres címke között. (További információ az "Üres hely egy bekeretezett sor vagy oszlop körül" című résznél, a 631. oldalon található.)

nyomógomb

```
: button {
  action  alignment  fixed_height  fixed_width
  height  is_cancel  is_default  is_enabled
  is_tab_stop  key  label  mnemonic  width
}
```

A nyomógomb zóna egy valóságos nyomógombra hasonlít. A nyomógomb `label` attribútuma a nyomógombra írt szöveget adja meg. A nyomógombok a felhasználó számára azonnal látható akciókhoz megfelelőek: kilépés egy párbeszédpanelből, belépés egy beágyazott párbeszédpanelbe, és így tovább.

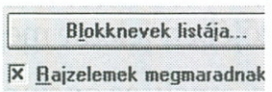
A párbeszédpanelnek tartalmaznia kell egy OK nyomógombot (vagy annak megfelelőjét), amelyet a felhasználó a panel használata (vagy elolvasása) után megnyom. Sok párbeszédpanel egy Mégsem nyomógombot is tartalmaz, amely lehetővé teszi, hogy a felhasználó bármilyen változtatás nélkül elhagyhassa a párbeszédpanelt.

A párbeszédpaneleknek a "Párbeszédpanelek kilépő nyomógombjai és hibazóna" című résznél, a 667. oldalon leírt előre definiált kilépő nyomógomb-részösszeállításokat kell használniuk. Ezek a részösszeállítások garantálják, hogy az ebben a fejezetben leírt attribútumok értékei megfelelőek.

Megjegyzés Amennyiben az alapértelmezés szerinti nyomógomb és az mégsem nyomógomb megegyezik, meg kell bizonyosodnia róla, hogy legalább még egy kilépő nyomógombhoz hozzá van rendelve a `done_dialog` függvényt meghívó akció. Különben a párbeszédpanelt mindig csak elvetni tudja.

`label` Megadja a nyomógombon megjelenő szöveget.

oszlop

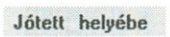



```
: column {
  alignment  children_alignment
  children_fixed_height  children_fixed_width
  fixed_height  fixed_width  height  label  width
}
```

Az oszlopban a zónák függőlegesen helyezkednek el, a DCL fájlban található felsorolásuk sorrendjében. Az oszlop bármilyen zónát tartalmazhat (az önmagában álló rádiógomb kivételével), *belértve* sorokat és más oszlopokat is.

A keret nélküli oszlopnak a szabvány elrendezési attribútumokon kívül nincs más attribútuma.

szöveglánc




```
: concatenation {
}
```

A szöveglánc olyan, szövegből álló sor, amely több `text_part` zóna összefűzéséből keletkezett. Ez akkor hasznos, ha a futási idő alatt változó szöveget kíván beilleszteni egy állandó üzenetbe. Az egész `concatenation` zónát szegélyező tér veszi körül.

A `concatenation` zóna a *base.dcl* fájlban van definiálva. A `concatenation` használatát a "bekezdés" című résznél, a 676. oldalon található példa szemlélteti.

párbeszéd

```
: dialog {
  initial_focus  label  value
}
```

A párbeszéd a párbeszédpanelt definiáló zóna. *Ne* adja meg egyszerre a `label` és `value` attribútumokat: a `value` attribútum felülírja a `label` értékét.

`label` Megadja az opcionális címet, amelyet a keret tetején vagy a párbeszédpanel cím mezőjében jelenít meg.

`value` Egy olyan karakterláncot ad meg, amely az opcionális párbeszédpanel címét jeleníti meg. Az elrendezéskor azonban nem vizsgálja meg az értékét, ezért ha ilyen módon adja meg a címet, győződjön meg róla, hogy a párbeszédpanel elég széles; különben a rendszer csonkolhatja a szöveget .

A `dialog` zóna esetében az elrendezési szempontoktól eltekintve a `label` és `value` attribútumok egymás megfelelői. A cím futásidő alatti megváltoztatásához használja a 13. fejezetben ismertetett **`set_tile`** függvényt.

`initial_focus` A zóna azon `key` attribútumát adja meg, amely a kezdeti billentyűzetfókuszt fogadja.

szerkesztőablak

```
: edit_box {
  action  alignment  allow_accept  edit_limit
  edit_width  fixed_height  fixed_width  height
  is_enabled  is_tab_stop  key  label  mnemonic
  value  width
}
```

A szerkesztőablak olyan mező, amelybe a felhasználó egysoros szöveget írhat, vagy szerkeszthet. Egy opcionális `címke` jelenhet meg a mező bal oldalán. Ha a bevitt szöveg hosszabb a szerkesztőablaknál, vízszintesen görgethető.

Egyes grafikus felhasználói felületek esetében a szerkesztőablaknak megfelelő zóna neve szövegablak (*text box*), szövegmező (*text field*) vagy szerkesztőmező (*edit field*).

Ha a `címkét` balra, a szerkesztőablakot pedig jobbra igazítja könnyebb az `edit_box` zónákat függőlegesen eligazítani.

`label` Címként jelenik meg. Ha meg van adva, a címke az `edit_box` zónán belül balra igazítva jelenik meg.

`value` A kezdeti ASCII érték, ami balra igazítva jelenik meg a szerkesztőablak szerkesztő (adatbeviteli) részében. A szerkesztőablakban található érték végén egy nulla karakter (`\0`, vagy az ADS nyelvben `EOS`) áll. Ha a felhasználó több karaktert ír be, mint amit az `edit_limit` megenged, a karakterlánc csonkolódik, a végére pedig a nulla karakter kerül.

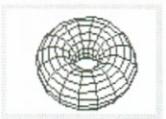
hibazóna

```
errtile;
```

A hibazóna a párbeszédpanel alján megjelenő szövegzóna. Alapértelmezés szerint üres, de a programok megjeleníthetnek üzeneteket benne, ha beállítják az `"error"` kulcsú zóna értékét.

Az `errtile` zóna a *base.dcl* fájlban van definiálva.

ábra



```
: image {
  action  alignment  aspect_ratio  color
  fixed_height  fixed_width  height  is_enabled
  is_tab_stop  key  label  mnemonic  value  width
}
```

Egy ábra olyan négyszög, amelyben vektor grafikus kép jelenik meg. Az ábrák ikonok, vonaltípusok, szövegfontok és színminták megjelenítésére használatosak az AutoCAD párbeszédpaneljeiben. Az "Ábrák létrehozása" című résznél, a 320. oldalon megtalálja, hogyan generálhat ábrákat az ábrazónák számára.

Az ábrazónához explicit módon megadott `width` és `height` attribútumot, vagy ezek egyikét és egy `aspect_ratio` attribútumot kell hozzárendelni.

ábra-nyomógomb

```
: image_button {
  action alignment allow_accept aspect_ratio
  color fixed_height fixed_width height
  is_enabled is_tab_stop key label mnemonic
  width
}
```

Az ábra-nyomógomb olyan zóna, amelyen szöveges címke helyett grafikus ábra jelenik meg.

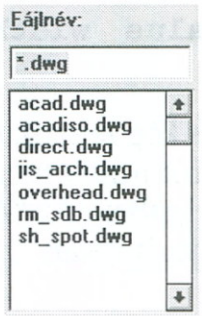
Egyes grafikus felhasználói felületek esetében az ábra-nyomógombnak megfelelő zóna neve ikon (*icon*), kép (*picture*) vagy hieroglifa (*glyph*).

Amikor a felhasználó kiválaszt egy ábra-nyomógombot, a program megkapja a kiválasztott pont koordinátáit. Ez akkor hasznos, ha egy kicsinyített rajzot kíván megjeleníteni, és annak különböző régióihoz különböző jelentést rendel hozzá.

Az "Ábrák létrehozása" című résznél, a 320. oldalon megtalálja, hogyan generálhat ábrákat az ábrazónák számára.

Az ábra-nyomógombhoz explicit módon megadott `width` és `height` attribútumot, vagy ezek egyikét és egy `aspect_ratio` attribútumot kell hozzárendelni.

listaablak




```
: list_box {
  action alignment allow_accept fixed_height
  fixed_width height is_enabled is_tab_stop
  key label list mnemonic multiple_select tabs
  value width
}
```

A listaablak sorokba rendezett szöveges karakterláncokból álló lista. A lista hossza változó, a listaablakok hosszúsága azonban rögzített lehet, amennyiben egy másfajta zóna -- például rádiógombok csoportja -- túl sok helyet foglal el a párbeszédpanelből. Amikor a felhasználó kiválaszt egy sort, az kiemelve jelenik meg. A listaablak több sort is tartalmazhat, mint amennyi az ablakban elfér; ezért a listaablak jobb oldalán mindig megjelenik egy görgetősáv. (A görgetősáv csak akkor engedélyezett, ha a lista több tételt tartalmaz, mint amennyi egyszerre látható.) A görgetősáv kurzorának vontatásával vagy a nyilakra történő kattintással a felhasználó végignézheti a listaablak tételeit. Egyes alkalmazások több sor kiválasztását is megengedik.

Egyes grafikus felhasználói felületek esetében a listaablaknak megfelelő zóna neve görgetőablak (*scroll box*) vagy görgethető lista (*scrolling list*).

A listák kezelését a "Listaablakok és legördülő listák" című résznél, a 317. oldalon ismertetjük.

`label` A listaablak fölött megjelenő szöveg.

`value` Idézőjeles karakterlánc, amely nullát (`" "`) vagy több, szóközzel elválasztott, egész számot tartalmaz (nincs alapértelmezés szerinti értéke). Minden egész szám egy nulláról induló index, amely a lista kezdetben kiválasztott tételét jelöli. Ha a `multiple_select` attribútum értéke `false`, a `value` attribútum nem tartalmazhat egynél több számot.

Ha a value karakterlánca üres (`" "`), kezdetben a lista egyik tétele sincs kiválasztva. Ebben az esetben nincs is szükség ennek az attribútumnak a megadására.

ok

`ok_only;`

Az `ok_only` zóna olyan önmagában álló OK nyomógomb, amilyet a figyelmeztető ablakok használnak. Az OK nyomógomb key attribútuma `"accept"`.

Az `ok_only` zóna a *base.dcl* fájlban van definiálva.

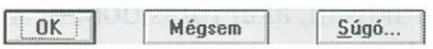
ok, mégsem

`ok_cancel;`

Az `ok_cancel` zóna az OK és Mégsem nyomógombok kombinációja. Megtalálható minden olyan párbeszédpanel esetében, amely adatok megváltoztatására szolgál. A Mégsem nyomógomb key attribútuma `"cancel"`.

Az `ok_cancel` zóna a *base.dcl* fájlban van definiálva.

ok, mégsem, súgó



`ok_cancel_help;`

A zóna az `ok_cancel` csoport és a Súgó nyomógomb kombinációja. A Súgó nyomógomb key attribútuma `"help"`. A Súgó nyomógombokat

az alkalmazások fő párbeszédpaneljében és egyéb bonyolult párbeszédpanelekben érdemes elhelyezni. A Súgó nyomógombot kezelő függvény az **acad_helpdlg** hívásával képes megjeleníteni az előre definiált AutoCAD Súgó párbeszédpanelt. Az **acad_helpdlg** függvény ismertetése az 5.,"Az állapotsor konfigurálása és a DIESEL karakteres kifejezésnyelv" című fejezetben található.

Az `ok_cancel_help` zóna a *base.dcl* fájlban van definiálva.

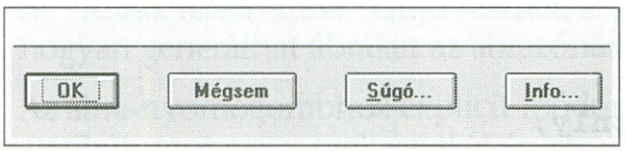
ok, mégsem, súgó, hibazóna

```
ok_cancel_help_errtile;
```

Az `ok_cancel_help_errtile` zóna kényelmes lehetőség arra, hogy egyszerre adja meg a kilépő nyomógombot és a hibazónát.

Az `ok_cancel_help_errtile` zóna a *base.dcl* fájlban van definiálva.

ok, mégsem, súgó, info

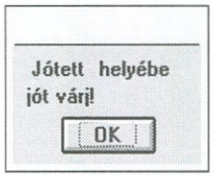



```
ok_cancel_help_info;
```

Az `ok_cancel_help_info` zóna mindent elvégez, amit az `ok_cancel_help` zóna, de egy információs nyomógombot is tartalmaz, további információk megjelenítése céljából. Megjelenítheti az alkalmazás nevét, cégének emblémáját, az alkalmazás verziószámát, a segítség elérésének módját, és így tovább. Az Info nyomógomb key attribútuma `"info"`.

Az `ok_cancel_help_info` zóna a *base.dcl* fájlban van definiálva.

bekezdés




```
: paragraph {
}
```

A bekezdés `text_part` vagy `concatenation` zónákból álló, függőleges elrendezésű csoport. Segítségével bekezdések szerkeszthetők folyó szövegből akár statikusan, akár futásidőben. A teljes `paragraph` zónát szegély veszi körül.

A `paragraph` zóna *base.dcl* fájlban van definiálva.

A lap szélén látható illusztráció például a következő DCL felhasználásával készült:

```
: paragraph {
  : concatenation {
    : text_part {
      label = "Jótett ";
    }
    : text_part {
      label = "helyébe";
    }
  }
  : text_part {
    label = "jót várj!";
  }
}
```

legördülő lista

```
: popup_list {
  action alignment edit_width fixed_height
  fixed_width height is_enabled is_tab_stop
  key label list mnemonic tabs value width
}
```

A legördülő lista funkcionálisan megegyezik a a listaablakkal. Amikor a párbeszédpanel először jelenik meg, a legördülő lista "összecsukott" állapotban van, és a képernyőn egy nyomógombra hasonlít, azzal a különbséggel, hogy a jobb oldalán egy lefelé mutató nyíl látható. Amikor a felhasználó rákattint a szövegre vagy a nyílra, a lista legördül, és további választási lehetőségeket jelenít meg. (Néhány platform esetében a felhasználónak a nyílra kell kattintania, nem a szövegre.) A legördült lista jobb oldalán görgetősáv található, amely a listaablak görgetősávjához hasonlóan működik. Összecsukott állapotban az aktuálisan kiválasztott tétel látható a legördülő lista mezőjében. A legördülő listák nem engedik meg több tétel egyidejű kiválasztását.

Egyes grafikus felhasználói felületek esetében a legördülő listának megfelelő zóna neve kombinációs ablak (*combination box, combo box*) vagy lehulló listaablak (*drop-down list box*).

A listák kezelését a "Listaablakok és legördülő listák" című résznél, a 317. oldalon ismertetjük.

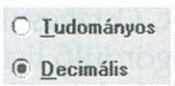
Egyes platformok figyelmen kívül hagyják a legördülő lista `height` attribútumát, és ugyanazt a rögzített magasságot rendelik az összes összecsukott állapotban lévő legördülő listához.

`label` Címként jelenik meg a legördülő lista bal oldalán. Ha meg van adva, a `label` a `popup_list` zóna szélességén belül balra igazítva helyezkedik el.

`edit_width` A lista szöveges részének karakterszélesség-egységekben kifejezett szélességét adja meg. Ebbe nem számít bele a bal oldalt látható esetleges címke, sem pedig a jobb oldalt elhelyezkedő nyíl (vagy görgetősáv). Ha az `edit_width` nincs megadva, vagy nulla, és a zóna szélessége nem rögzített, akkor a mező kitölti a rendelkezésére álló helyet. Értéke lehet egész vagy valós szám. Ha az `edit_width` nullától különböző értékű, a mező a zóna által elfoglalt helyen belül jobbra igazítva helyezkedik el. Ha elrendezési okokból a zónát tágítani kell, akkor a PDB szolgáltatás üres helyet illeszt a `label` és a mező szerkesztési része közé.

`value` Idézőjeles karakterlánc, amely egy egész számot tartalmaz (alapértelmezése: `"0"`). Ez az egész szám egy nulláról induló index, amely a lista aktuálisan kiválasztott tételét jelöli (azt, amelyik a zóna összecsukott állapotában látható).

rádiógomb




```
: radio_button {
  action  alignment  fixed_height  fixed_width
  height  is_enabled  is_tab_stop  key  label
  mnemonic  value  width
}
```

A rádiógomb a *rádióoszlopot* vagy a *rádiósort* alkotó nyomógombok csoportjának egyik eleme. Ezek úgy működnek, mint egy autórádió nyomógombjai: egyszerre csak egyikük lehet kiválasztva, és amikor az egyiket megnyomja, akkor az oszlopnak (vagy sornak) az a nyomógombja, amely addig be volt nyomva, kiugrik. Az opcionális `label` attribútum értéke a rádiógombtól jobbra jelenik meg. A PDB szolgáltatás hibát jelez, ha egy rádiógombot rádióoszlopon vagy rádiósoron kívül próbál elhelyezni.

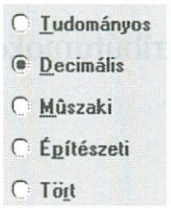
`label` A rádiógombtól jobbra (vagy néhány platform esetében annak felületén) megjelenő szöveg .

`value` Idézőjeles karakterlánc (nincs alapértelmezés szerinti értéke). Ha a `value "1"`, a `radio_button` bekapcsolt;

ha `"0"`, a `radio_button` kikapcsolt állapotban van; minden egyéb érték `"1"`-gyel egyenértékű.

Ha valamilyen véletlen folytán egy rádiócsoportban egynél több olyan `radio_button` zóna lenne, amelyre nézve a `value = "1"`, akkor ezek közül csak az utolsó lesz bekapcsolva. (Ez egy DCL fájlban fordulhat elő. Miután a párbeszédpanel működésbe lépett, a PDB szolgáltatás kezeli a rádiógombokat, és biztosítja, hogy csoportonként csak egy legyen bekapcsolva.)

rádióoszlop

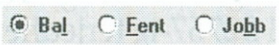



```
: radio_column {
  alignment  children_alignment
  children_fixed_height  children_fixed_width
  fixed_height  fixed_width  height  label  width
}
```

A rádióoszlop rádiógombokat tartalmaz, amelyek közül egyszerre csak egy lehet kiválasztva. Egy rádióoszlop az egymást kölcsönösen kizáró választási lehetőségek rögzített sorozatát jeleníti meg. A rádióoszlopokhoz -- az egyéb oszlopoktól eltérően -- akció is hozzárendelhető.

value Idézőjeles karakterlánc, amely az éppen kiválasztott rádiógomb `key` attribútumát tartalmazza (ennek értéke `"1"`).

rádiósor




```
: radio_row {
  alignment  children_alignment
  children_fixed_height  children_fixed_width
  fixed_height  fixed_width  height  label  width
}
```

A rádiósor --a rádióoszlophoz hasonlóan-- rádiógomb-zónákat tartalmaz, amelyek közül egyszerre csak egy lehet kiválasztva. rádiósorokhoz akció is hozzárendelhető.

value Idézőjeles karakterlánc, amely az éppen kiválasztott rádiógomb `key` attribútumát tartalmazza (ennek értéke `"1"`).

Megjegyzés A rádiósorokat nem olyan könnyű használni, mint a rádió-oszlopokat, mivel az egérnek nagyobb utat kell megtennie.

Akkor használjon rádiósorokat, ha az opciók száma kettő-négy, illetve ha a címkék rövidek.

sor

```
: row {
  alignment  children_alignment
  children_fixed_height  children_fixed_width
  fixed_height  fixed_width  height  label  width
}
```

A sor hasonló az oszlophoz, a zónák azonban nem függőlegesen, hanem vízszintesen helyezkednek el benne, a DCL fájlban található felsorolásuk sorrendjében.

A keret nélküli sornak a szabvány elrendezési attribútumokon kívül nincs más attribútuma.

csúszka

```
: slider {
  action  alignment  big_increment  fixed_height
  fixed_width  height  key  label  layout
  max_value  min_value  mnemonic  small_increment
  value  width
}
```

A csúszka számszerű érték kiválasztására szolgál. A felhasználó a csúszka mutatóját jobbra-balra (vagy fel-le) vontathatja, hogy beállítson egy értéket, amelynek jelentése az alkalmazástól függ. Ez az érték karakterláncként adódik vissza, amely egy meghatározott értéken belüli előjeles egész számot tartalmaz (az egész szám egy 16 bites érték, így a maximális tartomány –32768-tól 32767-ig terjedhet). Az alkalmazás ehhez tetszés szerinti skálát rendelhet hozzá.

Néhány platform esetében további szabályzógombok (például plusz és mínusz nyomógombok) segítségével a felhasználó rögzített növekménnyel is léptetheti a csúszkát. E növekmények értékét szintén Ön határozhatja meg.

Néhány grafikus felhasználói felület esetében a csúszkának megfelelő zóna neve számlap (*dial*).

value Idézőjeles karakterlánc, amely a csúszka aktuális (egész) értékét tartalmazza (alapértelmezése: `min_value`).

szöveg

Fájltípusok:

```
: text {
  alignment  fixed_height  fixed_width  height
  is_bold  key  label  value  width
}
```

A szövegzóna egy karakterlánc megjelenítésére szolgál, címadás vagy tájékoztatás céljából.

Mivel a zónák többsége saját `label` attribútumát használja címadásra, legtöbbször nincs szükség a szövegzónák használatára. Egy olyan szövegzónának azonban, amelyet általában üresen tart, jó hasznát veheti a felhasználói akciók visszajelzésére, hibaüzenetek, és figyelmeztetések megjelenítésére.

A figyelmeztető ablakokról és a hibazónákról a "Párbeszédpanelek kilépő nyomógombjai és hibazóna" című résznél, a 667. oldalon és a "Hibakezelés" című résznél, a 646. oldalon ejtünk szót.

Ha azt akarja, hgoy az üzenet állandó legyen, akkor a `label` attribútumban adja azt meg, és ne adja meg a `width` és `value` attribútumokat. Amennyiben az üzenetet a futásidőben meg akarja változtatni, adja meg azt a `value` attribútumban, és egy olyan értékű `width` attribútumot rendeljen hozzá, hogy a `value` attribútumban elhelyezni szándékozott karakterlánc beleférjen. Miután a párbeszédpanel elrendezése megtörtént, zónáinak méretét nem lehet változtatni, ezért ha Ön a **`set_tile`** függvény segítségével az adott szélességnél hosszabb karakterláncot ad meg, a szöveg megcsonkítva jelenik meg.

`label` A megjelenített szöveg. Elrendezésekor a `text` zóna szélessége az az érték lesz, amely a `width` attribútumában (ha az meg van adva a DCL fájlban) szereplő, illetve a `label` attribútuma (ha adott) által megkövetelt szélességi értékek közül a nagyobbik. Ezek közül legalább az egyiket meg kell adni.

`value` A `label` attribútumhoz hasonlóan a `value` is egy, a text zónában megjeleníteni kívánt karakterláncot ad meg. Ennek azonban a zóna elrendezésére nincs hatása.

szövegrész

Elhelyezés:

```
: text_part {
  label
}
```

A szövegrész olyan szövegzóna, amely egy nagyobb szövegdarab alkotórésze. A `text_part` zóna szegélye ki van iktatva, így más `text_part` zónákkal szöveglánc- vagy bekezdés-zónává egyesíthető.

A `text_part` zóna a *base.dcl* fájlban van definiálva. A `text_part` zóna használatára a "bekezdés" című résznél, a 676. oldalon talál példát.

kapcsoló

```
: toggle {
  action  alignment  fixed_height  fixed_width
  height  is_enabled  is_tab_stop  label  width
}
```

A kapcsoló egy Boolean értéket (`"0"` vagy `"1"`) vezérel. Kisméretű négyszögként jelenik meg, tőle jobbra egy esetleges címkével. Amikor a felhasználó kiválasztja a négyszöget, egy ellenőrző jel (pipa) vagy X jelenik meg (illetve tűnik el) benne. A kapcsolók lehetővé teszik, hogy a felhasználó megtekintse és megváltoztassa a be/ki opciók állapotát. A kapcsolók ki/be mező (*check boxes*) néven is ismertek.

label A kapcsoló négyszögétől jobbra (néhány platform esetében azon belül) megjelenő szöveg.

value Idézőjeles karakterlánc, amely egy egész számot tartalmaz (alapértelmezése: `"0"`), és megadja a `toggle` zóna kezdeti értékét. Ha a karakterlánc értéke `"0"`, a kapcsoló négyszöge üres (ellenőrző jel nélküli). Ha `"1"`, a négyszögben egy pipa (vagy egy *X*) látszik.

helykitöltő (spacer)

```
: spacer {
  alignment  fixed_height  fixed_width
  height  width
}
```

A helykitöltő egy üres zóna. Csak elrendezési célokra, a szomszédos zónák méretének és elhelyezkedésének befolyásolására használatos. Mivel a PDB szolgáltatás az üres helyeket automatikusan kezeli, csak különleges esetben használja. Így biztosítható a párbeszédpanelek egységes megjelenése. További információ a "DCL technikák" című résznél, a 629. oldalon található.

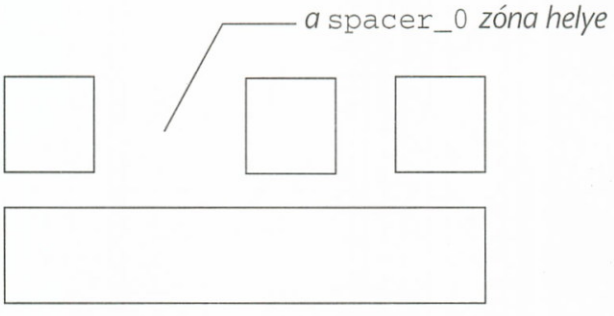
A `spacer` zónának a szabvány elrendezési attribútumokon kívül nincs más attribútuma.

helykitöltő (spacer_0)

`spacer_0;`

Az alábbi ábrán bemutatott `spacer_0` zóna olyan helykitöltő, amelynek általában nincs szélessége. Megjelöl azonban egy pontot a zóna csoportban, ahová Ön helyet kíván beilleszteni, ha a csoportot szét kell húzni az elrendezés során. Ha a `spacer_0` zónák csoportjához pozitív szélességet rendel, valamennyien azonos nagyságú helyet foglalnak el a térkitöltés során.

A `spacer_0` zóna a *base.dcl* fájlban van definiálva.



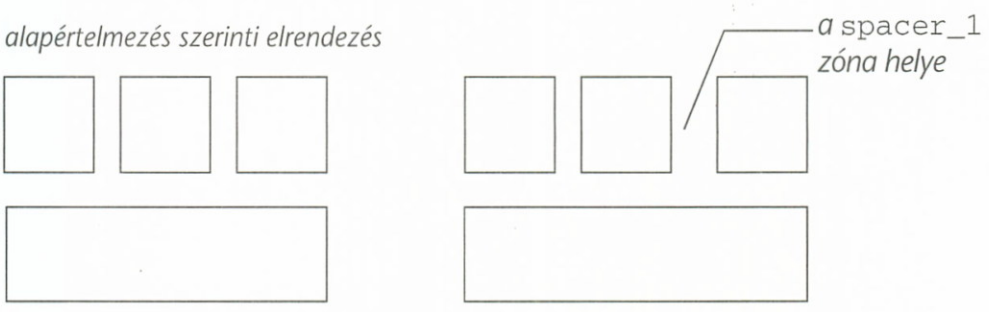
A spacer_0 zóna használata

helykitöltő (spacer_1)

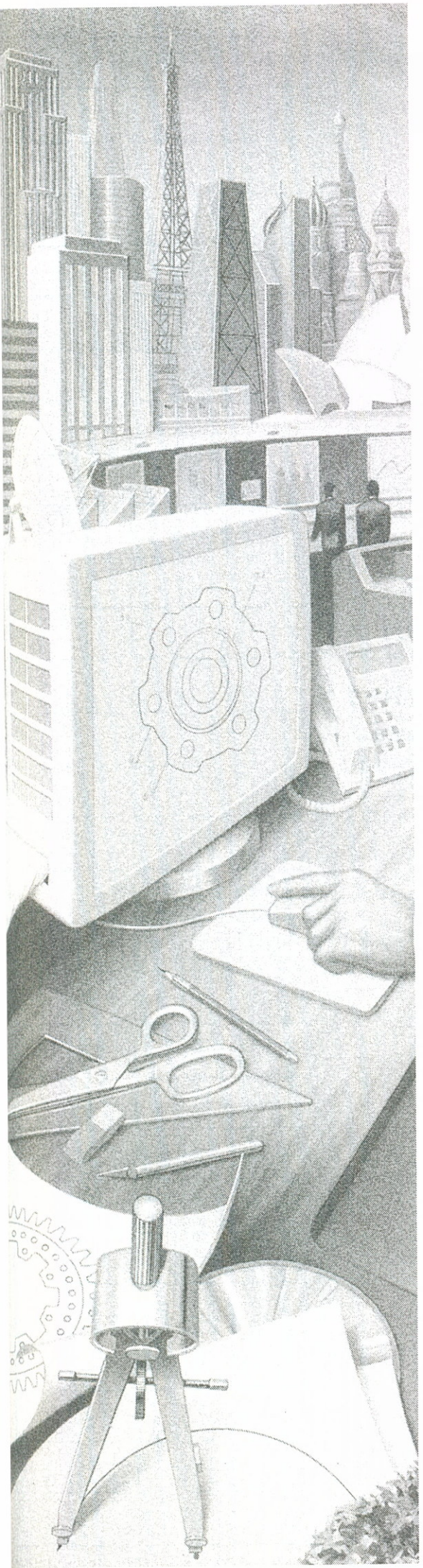
`spacer_1;`

Az alábbi ábrán bemutatott `spacer_1` zóna olyan helykitöltő, amelynek szélessége és magassága is egy. A legkisebb olyan helykitöltőként használatos, amely még látható a felhasználó számára.

A `spacer_1` zóna a *base.dcl* fájlban van definiálva.



A spacer_1 zóna használata



FÜGGELÉK

TARTALOM

A. FÜGGELÉK ■ ASCII KÓDOK 689

B. FÜGGELÉK ■ FÁJLOK LISTÁJA 693

Részleges fájllista 694
Fájltípusok 697

A

ASCII KÓDOK

Ez a melléklet a szabvány ASCII kódokat mutatja be. Az oktális forma (alakja `\nnn`) a karakter- vagy karakterlánc-konstansoknál lehet hasznos. Az Ön által használt rendszer további kódokat is definiálhat a kibővített, 256-os karakterkészletben (az újabb kódok 127-nél nagyobbak). Egyes rendszerek a ritkán használt ASCII kódokat (1–6 illetve 14–26) is újra definiálják. Ha mind decimális, mind oktális formában látni kívánja operációs rendszerének karakterkészletét, használja az AutoLISP által definiált ASCII parancsot (melyet az "ASCII kódok konverziója" című résznél, a 246. oldalon ismertetünk), amely a képernyőre és az *ascii.txt* nevű fájlba írja azt.

Dec.	Okt.	Hex.	Karakter
0	000	00	NUL
1	001	01	SOH
2	002	02	STX
3	003	03	ETX
4	004	04	EOT
5	005	05	ENQ
6	006	06	ACK
7	007	07	BEL (bell)
8	010	08	BS (backspace)
9	011	09	HT
10	012	0A	LF (line-feed)
11	013	0B	VT
12	014	0C	FF
13	015	0D	CR (return)
14	016	0E	SO
15	017	0F	SI
16	020	10	DLE
17	021	11	DC1
18	022	12	DC2
19	023	13	DC3
20	024	14	DC4
21	025	15	NAK
22	026	16	SYN
23	027	17	ETB
24	030	18	CAN
25	031	19	EM
26	032	1A	SUB
27	033	1B	ESC (escape)
28	034	1C	FS
29	035	1D	GS
30	036	1E	RS
31	037	1F	US

Dec.	Okt.	Hex.	Karakter
32	040	20	(space)
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	-
46	056	2E	.
47	057	2F	/
48	060	30	0
49	061	31	1
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?

Dec.	Okt.	Hex.	Karakter
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	^
95	137	5F	_

Dec.	Okt.	Hex.	Karakter
96	140	60	‘
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~
127	177	7F	DEL (delete)

B

FÁJLOK LISTÁJA

Ez a melléklet az AutoCAD program- és kiegészítő fájljainak részleges listáját, valamint az AutoCAD által használt fájlok típusát (kiterjesztését) mutatja be.

Részleges fájllista

Az ebben a részben felsorolt fájlok általános áttekintést nyújtanak az AutoCAD program- és kiegészítő fájljairól, illetve azok használatáról. Ez a lista nem tartalmazza AutoCAD programcsomaghoz tartozó összes fájlt. A *fájlnév* (*.exe*) jelölésű fájlok kiterjesztése platformonként változik.

Az AutoCAD részleges fájllistája

Fájl neve	*Leírás*
acad (.exe)	Az AutoCAD futtatható fájlja
acad.ads	Megadja az automatikusan betöltött ADS alkalmazásokat
acad.ahp	Platformtól független AutoCAD súgófájl
acad.ase	ASE kiegészítő fájl
acad.cfg	Konfiguráció (az *acad.cfg* fájllal azonos könyvtárban található *acad.bak* fájl az elmentett konfigurációs fájl)
acad.dcl	Az AutoCAD parancsok által használt párbeszédpanel-definíciós fájl
acad.dwg	Szabvány prototípus rajzok
acad.err	Hiba naplófájl
acad.hlp	Windows súgófájl
acad.ini	Inicializáló fájl
acad.lin	Szabvány vonaltípus-könyvtár fájl
acad.mln	Szabvány sokvonal-könyvtár fájl
acad.mnl	Szabvány menü AutoLISP fájl
acad.mnu	Szabvány menüsablon fájl
acad.msg	Az NÉVJEGY parancs által megjelenített üzenet
acad.pat	Szabvány sraffozási könyvtárfájl
acad.pgp	Programparaméter-fájl
acad.psf	AutoCAD PostScript fontleképző fájl
acad.rx	Az automatikusan betöltött ARX alkalmazásokat adja meg
acad.slb	Dia könyvtárfájl
acad.unt	Mértékegységek konverziós fájlja

Az AutoCAD részleges fájllistája (*folytatás*)

Fájl neve	*Leírás*
acadapp (.exe)	AutoCAD alkalmazások programfájlja (ADS alkalmazás)
acadl.exp	AutoLISP értelmező (csak DOS esetén)
acadps (.exe)	AutoCAD PostScript értelmező (ADS alkalmazás)
acadps.ps*	AutoCAD PostScript értelmező kiegészítő fájlok
acadr13.lsp	Automatikusan betöltött AutoLISP segédfájlok
*acis.**	ACIS kiegészítő fájlok
actutdde.dll	Súgófájl az AutoCAD DDE csatoláshoz (csak Windows esetén)
ai_utils.lsp	AutoLISP segédfájl (az *acadr13.lsp* fájl használja)
ase.**	AutoCAD SQL felület fájlok
asi.**	AutoCAD SQL felület fájlok
ase.arx	AutoCAD SQL felület program (ARX alkalmazás)
base.dcl	Fő párbeszédpanel definíciós fájlja
cfig386.exe	Kibővítő kapcsolót vezérlő program (csak DOS esetén)
*cfigphar.**	Kibővítő kapcsolót vezérlő kiegészítő fájlok (csak DOS esetén)
copying.gs	Ghostscript copyright információ
dg.dll*	Digitalizáló meghajtó fájlok (csak Windows esetén)
dg.exp*	Digitalizáló meghajtó fájlok (csak DOS esetén)
ds.dll*	Megjelenítő eszközt meghajtó fájlok (csak Windows esetén)
ds.exp*	Megjelenítő eszközt meghajtó fájlok (csak DOS esetén)
*dxfix.**	Dxfix alkalmazás fájlok
geom3d (.exe)	Háromdimenziós geometriai mozgás, ADS alkalmazás
geomcal (.exe)	Geometriai számológép, ADS alkalmazás
mc.exe	Menüfordító az AutoCAD számára
mvsetup.dfs	Az MNBEÁLL parancs alapértelmezés szerinti értékeinek fájlja

Az AutoCAD részleges fájllistája (*folytatás*)

Fájl neve	*Leírás*
*nullsurf.**	Render üres felület fájlok
pharlap.386	Eszközfálj a Windows 3.1 használatához
pl.dll*	Plotter meghajtó fájlok (csak Windows esetén)
pl.exp*	Plotter meghajtó fájlok (csak DOS esetén)
quiktour.tbk	Toolbook alkalmazásfájl (csak Windows esetén)
*r13new.**	*Újdonságok* programfájl (csak DOS esetén)
rasterin (.exe)	Rasterin ADS alkalmazás
rc.exp*	Renderelő kombinált megjelenítő eszközt meghajtó fájlok (csak DOS esetén)
*readme.**	Readme dokumentumok
*render.**	AutoCAD Render kiegészítő fájlok
render.arx	AutoCAD Render alkalmazásfájl
rh.exp*	Render hard-copy meghajtó fájlok (csak DOS esetén)
shroom.com	Segédprogram, amely megnöveli a memóriát a SHELL parancs számára
shroom.doc	Információ a *shroom.com* programról
slidelib.exe	Diakönyvtár fordító
tbk.**	Toolbook programkiegészítő fájlok(csak Windows esetén)
tbook.exe	Toolbook programfájl (csak Windows esetén)
*tutorial.**	Tankönyv fájlok (csak Windows esetén)
uglyr.gsf	A PostScript értelmező által használt fontfájl, amely az ismeretlen fontok helyettesítésére szolgál
whatsnew.tbk	Toolbook alkalmazásfájl (csak Windows esetén)

Fájltípusok

A következő táblázat felsorolja az AutoCAD programcsomaghoz biztosított fájlok típusát (bizonyos kiterjesztéssel rendelkező fájlokat).

AutoCAD fájltípusok

Fájl kiterjesztése	*Leírás*
.3ds	3D Studio fájl
.adt	Rajzellenőrzés eredményét jelentő fájlok
.ahp	AutoCAD súgófájl
.arx	ARX alkalmazásfájlok
.asm	ADS minta forráskód fájlok
.bak	Rajz háttérfájlok
.bat	Batch fájlok
.bdf	Videó megjelenítő fontfájlok (csak DOS esetén)
.bdf	VESA fontfájlok
.bkn	Vészhelyzetben használt háttér fájlok, amelyek a következő egyedi név szerint számozódnak (*n* minden fájl esetében növekszik)
.c	ADS forráskód fájlok
.cc	ADS forráskód fájlok
.cfg	Konfigurációs fájlok
.cus	Saját szótárfájlok
.dbf	Adatbázis fájlok az ASE tankönyv számára
.dcc	Párbeszédpanel színvezérlő fájlok (csak DOS esetén)
.dce	Párbeszédpanel hibajelentő fájlok
.dcl	Párbeszédpanel-definíciós fájlok
.dct	Szótárfájlok
.def	ADS forráskód fájlok
.dll	Platform-specifikus, dinamikusan csatolt könyvtárak
.dwg	Rajzfájlok

AutoCAD fájltípusok (*folytatás*)

Fájl kiterjesztése	*Leírás*
.dxb	Rajzok közötti bináris cserefájlok
.dxf	Rajzok közötti cserefájlok (ASCII vagy bináris)
.dxx	Attribútumkivonat fájlok (DXF formátum)
.eps	Befoglalt PostScript fájlok
.err	Az AutoCAD program összeomlása esetén a hibalistát tartalmazza
.exp	Meghajtó és ADS futtatható fájlok (csak DOS esetén)
.fnt	Videó megjelenítő fontfájlok (csak DOS esetén)
.h	ADS include fájlok
.hdx	Súgó index fájlok
.hlp	Windows súgófájlok
.lib	ADS könyvtárfájlok
.lin	Vonaltípus könyvtárfájlok
.lsp	AutoLISP alkalmazásfájlok
.mid	Azonosító információs fájlok
.mli	AutoCAD Render anyag könyvtárfájlok
.mln	Sokvonal könyvtárfájlok
.mnc	Lefordított menüfájlok (csak Windows esetén)
.mnd	Menüdefiníciós fájlok (az mc futtatható fájlhoz)
.mnl	AutoLISP menüfájlok
.mnr	Menü forrásfájlok (csak Windows esetén)
.mns	Menü forrásfájlok (csak Windows esetén)
.mnu	Menü sablonfájlok
.mnx	Lefordított menüfájlok (csak DOS esetén)
.old	A konvertált rajzfájl eredeti változata
.pat	Sraffozási minta könyvtárfájlok
.pcp	Plot konfigurációs paraméterfájl

AutoCAD fájltípusok (*folytatás*)

Fájl kiterjesztése	*Leírás*
.pfb	PostScript fontfájl
.pfm	PostScript metrikus fontfájlok
.plt	Plot kimeneti fájlok
.sab	Szabvány ACIS bináris fájlok
.scr	Parancs forgatókönyv fájlok
.shp	AutoCAD alak/font forrásfájlok
.shx	Lefordított alak/font fájlok
.slb	Diakönyvtár fájlok
.sld	Diafájlok
.stl	Sztereólitográfia fájlok
.ttf	TrueType fontfájlok
.txt	Attribútumkivonat vagy sablon fájlok (CDF/SDF formátum)
.xlg	Külső hivatkozások naplófájljai
.xmx	Külső üzenetek fájljai

**OSNAP képernyőalmenü, 229
3D
  kihúzási vektor, 253
  pontok, 207, 484
    bővített adatban, 291
  távolság, 424, 446
3D Studio fájlok, 531
  importálás, 530
3DFACE rajzelemtípus,
    csoportkódok, 576
3DSBE parancs, 530
3DSKI parancs, 531
3DSOLID rajzelemtípus,
    csoportkódok, 576

A

ábradiák, 120
ábramenük
  menük, 229
ábra-nyomógombok, 322, 644
  visszajelzési indíték kód, 308
ábrazóna
  menük, 119
    diák elkészítése, 123
ábrazónák, 319, 644
  függvény áttekintés, 370
  Lásd még AutoLISP
      függvények,
      menucmd
  menük
    címkék, 120
    szintaxis, 120
acad parancs, 158
acad.cfg AppData szakasza, 445,
    493
ACAD_GROUP rajzelemtípus,
    csoportkódok, 589
ACAD_MLINESTYLE rajzelemtípus,
    csoportkódok, 590
ACADCFG
  környezeti változók, 21
acadpsd.ps fájl, 83
adatbázis-objektumok
  elérési útvonal (ASE), 259
  hivatkozás (ASE), 260
  nevek (ASE), 259
adatbázis-sorszám, bővített
    adatban, 291
adatbevitel
  érvényesség, 238
  felhasználói bevitelre szolgáló
      függvények
    áttekintés, 357
  get, 330
adatbevitel lezárások, 159
adatbeviteli eszközök, 232
adattípusok
  ASE, 259
  és változók, 196
ADS, 157, 165, 191
  alkalmazások, 191
    automatikus betöltés,
      167
    betöltés, 166, 220, 526
    memóriából való törlés,
      527
    mint külső függvények,
      165
    pillanatnyilag betöltött
      alkalmazások
      listája, 381
    PostScript értelmező
      program, 82
  könyvtár és fejléc fájlok, 166
  párbeszédpanelek vezérlése,
      617
  programozási nyelv, 157
ads könyvtár, 166
ADS-definiált parancsok
  és hibakezelés, 218
affin transzformáció, 259
akciókifejezés változók
  $data, 306, 322
  $key, 306
  $reason, 306, 307
  $value, 306
  $x, 306
  $y, 306
  newtile, 307
aláhúzás (_)
  honosított változatok
      parancsai, 225,
      417
alak- és fontfájlok, fordítás, 50
ALAK parancs, 49
alak, rajzolás, 333
alakdefiníciós fájl, 50
  speciális kódok
    függőleges szövegre
      vonatkozó
      parancs, 60
    ív törtrésze, 58
    kihajlásával megadott ív,
      59
    nyolcad ív, 57
    X-Y elmozdítások, 56
  alul- és felülírás létrehozása,
      74
  maximális karakterek, 51
  speciális kódok, 54
    alak vége, 55
    alárendelt alak, 56
    méretbeállítás, 55

pozíció elmentése és visszaállítása, 55
rajzmód vezérlés, 55
vektorhossz- és iránykódok, 53
alapértelmezés szerinti akciók, 305, 309
alapmenük, 89, 127
alapmértékegységek, 249
alapvektorok, 254
"'alkalmazás' alkalmazás már be.van töltve" üzenet, 526
alkalmazáskezelő függvények, 218
ADS alkalmazások, 381, 526
alkalmazásnév bejegyzése, 293, 490
ARX alkalmazások, 385
asszociációs alkalmazáskezelt adatok a párbeszédpanel-zónáknál, 416
áttekintés, 354
Windows alkalmazás, 505
alkalmazásnevek, bővített adatban, 291
alkalmazás-specifikus adat, 322
alkönyvtárak
elválasztó jelek
/ (törtjel), 162
ALKTÖLT parancs, 161
állapot, menüpontok, 473
állapotsor
hosszúság, 154
konfiguráció, 140
konfiguráció MODEMACRO segítségével, 140, 154
szöveg írása, 458
álnév jellemző, AutoCAD parancsok definiálása, 213
általános LISP programozási nyelv. *Lásd* AutoLISP
általánosító megjegyzések(ASE), 262
ANGBASE rendszerváltozó, 234, 244, 496
ANGDIR rendszerváltozó, 234
anyagkönyvtár, 532
ANYAGTÁR parancs, 532
APERTURE rendszerváltozó, 239
AppData szakasz *acad.ini*, 227
APPID rajzelemtípus, csoportkódok, 590
APPID szimbólumtábla, 288, 293
arányténye zető, a bővített adatban, 293
ARC rajzelemtípus, 576
argumentumok
függvények, 215
szimbólumok mint, 215
választható, 215
aritmetikai függvények
áttekintés, 346
Lásd még egyedülálló függvények
ARX, 168
alkalmazások, 191, 529
betöltés, 220
felület, 157
definiált parancsok, és hibakezelés, 218
fájlok, betöltés, 411
ASCII
fájlok használata az adaptálás alatt, 18
függvények, 246, 408, 416, 488
kódok, 689
szövegfájlok AutoLISP alkalmazásokban, 247
szövegfájlok, AutoLISP alkalmazások tárolása, 161
ASE
AutoLISP felület, 259
beállítások, 387
csatolások
elérési útvonalak, 397
hozzáadás
azonosítók, 399, 402
kiválasztás, 407
felület függvényei, áttekintés, 362
hibakódok, 392
parancskészlet, 259
ASE parancs, 169
ASEADMIN parancs, 532
ASEEXPORT parancs, 533
ASELINKS parancs, 533
ASEROWS parancs, 533
ASESELECT parancs, 534
ASESQLED parancs, 534
ASEUNLOAD parancs, 169
ASI és AutoLISP adattípusok
adattípusok és ASI adattípusok, 261
asszociációs listák, 209
átalakító függvények
szögek, 328
Átmérő alparancs, 418
ATTDEF parancs, 198
ATTDEF rajzelemtípus, csoportkódok, 577
ATTEDIT parancs, 279
ATTRIB rajzelemtípus, csoportkódok, 578
attribútumok
blokkvég, 284
párbeszédpanel, 304
DCL érték visszaadása, 443
attribútumok következnek flag, vonalláncok, 284
AUNITS rendszerváltozó, 243
AUPREC rendszerváltozó, 243
"Az AutoCAD visszautasította a függvényt" hibaüzenet, 304, 417
AutoCAD
beviteli eszközök, 232
fájllista, 694
grafikus képernyő, 232
kiegészítő fájlok, 23
könyvtárszerkezet, 18
promptok, 159
AutoCAD futás ideje alatti kiterjesztés. *Lásd* ARX
AutoCAD parancsok
TÁBLA, 257
VISSZA, 217
AutoLISP, 157, 187, 189
alkalmazáskezelő függvények, 218
alkalmazások, 161
betöltés, 161
ASE felület, 259
AutoCAD parancsok definiálása, 213
AutoLISP kód betöltése, 160, 219
automatikus betöltés parancsa, 164

DIESEL kifejezések menükben, 144, 146, 148
előre definiált változók, 198
fájlok, betöltés, 411, 412
függvényszintaxis, 194
gyakorló példa, 327
használat menükben, 126
hibák, 163
hibakeresés, 514
hibakódok, 597, 598
hibaüzenetek, 597, 601
ASE, 262
belső hibák, 608
karakterlánc konverziós függvények, 243
karakterlánc-kezelés, 199
kifejezések
get*xxx*, 146
lekérdező és parancs függvények, 224
listakezelés, 206
megjegyzések, 161, 195, 328
memória, 472
memóriakezelés, 563
párbeszédpanelek vezérlése, 617
pontlisták, 207
programfájlok, 194
betöltés, 219
igazítás, 196
megjegyzések, 195
tabulálás, 196
rutinok automatikus betöltése és végrehajtása, 162
számok kezelése, 198
változók, 197
verziószám, 519
zárójelek () kifejezésekben, 192, 330
Lásd még DXF csoportkódok
AutoLISP alkalmazások fejlécei, 161
AutoLISP függvények
- (kivonás), 198, 250, 374
* (szorzás), 198, 374
error, 216, 438, 601
+ (összeadás), 190, 198, 250, 374
/ (osztás), 198, 250, 375
/= (nem egyenlő), 205, 376
< (kisebb, mint), 205, 376
<= (kisebb vagy egyenlő), 205, 376
= (egyenlő), 205, 375
> (nagyobb, mint), 205, 377
>= (nagyobb vagy egyenlő), 205, 377
^ (kalap), hatványozás, 250
~ (not), 198, 377
1- (csökkentés), 199, 378
1+ (növelés), 198, 378
abs, 198, 378
acad_colordlg, 224, 378
acad_helpdlg, 379
acad_strlsort, 191, 379
action_tile, 305, 307, 309, 311, 324, 340, 380
add_list, 317, 324, 380
ads, 218, 381
ads_command, 165
ads-ssget
bővített adatok, 572
alert, 216, 218, 381, 636, 646
alloc, 382, 566
and, 205, 382
Boole, 205, 412
logand, 470
angle, 238, 331, 382
angtof, 200, 246, 383
angtos, 200, 246, 384
append, 206, 385
apply, 210, 385
árkusz tangens, 409
arx, 218, 385
arxload, 169, 218, 386
arxunload, 169, 218, 386
ascii, 200, 246
ase_docmp, 387
ase_docurrent, 387
ase_dolinst, 260
ase_dolist, 388
ase_dopathcode, 389
ase_dopathmake, 390
ase_dopathname, 390
ase_dostatus, 391
ase_errcode, 392
ase_errdsc, 392
ase_errmsg, 392
ase_errqty, 393
ase_linkcreate, 393
ase_linkget, 395
ase_linkremove, 395
ase_linkupdate, 396
ase_lperase, 397
ase_lpisupdatable, 398
ase_lpkey, 398
ase_lplist, 398
ase_lpnew, 397
ase_lppath, 399
ase_lprename, 399
ase_lsadd, 399, 402
ase_lscmp, 400
ase_lscopy, 400
ase_lscreate, 401
ase_lsentsel, 402
ase_lserase, 403
ase_lsfree, 403
ase_lsget, 403
ase_lsintersect, 404, 406
ase_lsintersectfilter, 404
ase_lsisupdatable, 405
ase_lslpnames, 405
ase_lsmemb, 406
ase_lsqty, 406
ase_lsunion, 407
ase_lsxnames, 407
ase_version, 408
assoc, 206, 209, 277, 280, 408
atan, 198, 409
atof, 200, 246, 409
atoi, 200, 243, 311, 318, 410
atom, 210, 410
atoms_family, 210, 410, 565
autoarxload, 218, 411
autoload, 165, 218, 411, 412
autoxload, 218
beolvasás, 488
betöltés fájlokból, 219
Boole, 205, 412
boundp, 210, 414
C:XXX, 211
caddr, 208
cadr, 208
car, 206, 208, 209, 414
cdr, 206, 209, 414, 420
chr, 200, 243, 416
client_data_tile, 322, 324, 416
close, 254, 416
column_name, 260

command, 161, 163, 212, 213, 333, 416
és rajzelemnév-függvények, 274
cond, 205, 419
cons, 206, 209, 268, 420
cos, 198, 242, 420
csillag (*), szorzás, 198, 374
csökkentés (1-), 199, 378
cvunit, 243, 248, 420
defun, 163, 210, 219, 220, 421
vmon függvény és, 567, 568
dictnext, 288, 298, 300, 423
dictsearch, 288, 298, 300
dimx_tile, 319, 424
dimy_tile, 319, 424
distance, 331, 424
distof, 200, 243, 245, 425
done_dialog, 302, 314, 316, 324, 426
drawout, 333
drawtiles, 335
drow, 335
dtr, 328
egyenlő (=), 375
egyenlőség függvények, 205
end_image, 319, 324, 426
end_list, 317, 324, 427
entdel, 272, 274, 279, 298, 303, 427
szótár-rajzelemek eltávolítása, 300
entget, 252, 267, 272, 277, 279, 282, 427, 430
bővített adatok lekérdezése, 289
és globális karakter összehasonlítás, 204
és könyvtárobjektumok, 288, 298
és szimbólumtábla-bejegyzések, 298
entlast, 193, 272, 274, 284, 429
entmake, 252, 272, 282, 303, 430
és a bmake alkalmazás, 325
és szimbólumtáblák, 298
és szótárobjektumok, 300
entmod, 252, 272, 280, 281, 285, 287, 288, 303, 433
és szimbólumtáblák, 298
entnext, 272, 275, 286, 434
entsel, 235, 273, 275, 303, 435, 462
billentyűzet bevitel, 237
entupd, 272, 303, 434, 436
eq, 205, 437
equal, 205, 437
és
boolean, 268
eszközvezérlő függvények, 256
eval, 210, 439
exit, 216
Lásd még lezárás
exp, 198, 439
expand, 440, 566
expt, 198, 440
fájlkezelés, 228, 254
felhasználó által definiált függvények, 215
felhasználói bevitelre szolgáló függvények, 256
feltételes függvények, 205
fill_image, 319, 322, 324, 440
findfile, 254, 441
hivatkozásinév változó, 254
fix, 198, 442
float, 199, 442
foreach, 206, 442
gc, 266, 443, 564, 566, 567
gcd, 198, 443
geometriai segédfüggvények, 238
get_attr, 311, 324, 443
get_tile, 311, 324, 444
getangle, 234, 303, 444, 462
getcfg, 224, 445
getcorner, 234, 303, 445, 462
getdist, 234, 303, 446, 462
getenv, 224, 227, 234, 446
getfiled, 255, 447
defil változó, 255
getint, 234, 236, 303, 449, 462
getkword, 235, 303, 450, 462
getorient, 234, 303, 451, 462
getpoint, 232, 234, 303, 330, 451, 462
getreal, 232, 234, 237, 303, 452, 462
getstring, 232, 234, 303, 452, 462
getvar, 217, 224, 226, 234, 337, 453
get*xxx*, command függvény, 233, 417
gpuser, 330
graphscr, 227, 231, 303, 454
grclear, 227, 232, 303, 454
grdraw, 227, 303, 454
grread, 256, 303, 455
grtext, 227, 232, 303, 458
grvecs, 227, 232, 275, 303, 459
handent, 272, 274, 297, 460
hatványozás (^), 250
hatványozás. *Lásd* AutoLISP függvények, szorzás
help, 254, 461, 637
hibakezelés, 235, 392
hullám (~), not, 377
if, 205, 336, 461
initget, 212, 232, 235, 273, 462
bits argumentum, 236
strings argumentum, 236, 237
inters, 239, 466
itoa, 200, 243, 466
képernyővezérlés, 227
key_value, 261
kisebb vagy egyenlő (<=), 376
kisebb, mint (<), 376
kivonás (-), 198, 250, 374

konverziók, 243
ASCII, 246, 408, 416
egész számok, 410, 416, 442, 466, 492
hőmérséklet, 250
kisbetűre történő konvertálás, 507
koordináta-rendszerek, 251, 477, 479, 515
mértékegységek, 248, 420
nagybetűre történő konvertálás, 507
szögek, 246, 383
valós számok, 243, 250, 409, 425, 442, 492
könyvtárelérés, 298
lambda, 210, 467
last, 206, 467
legnagyobb közös osztó, 443
length, 206
link_id, 261
link_sel, 261
list, 206, 207, 268, 468
listp, 206, 468
load, 161, 162, 163, 218, 219, 469
load_dialog, 324, 340, 469, 625
log, 198, 470
logand, 198, 470
logior, 198, 471
lp_name, 260
lsh, 198, 471
mapcar, 206, 317, 471
max, 198, 472
mem, 472, 568
member, 206, 473
menucmd, 148, 227, 303, 473
min, 199, 475
minusp, 198, 475
mínusz jel (-), kivonás, 198, 250, 374
mode_tile, 310, 324, 476
nagyobb vagy egyenlő (>=), 377
nagyobb, mint (>), 377
namedobjdict, 298, 476
nem egyenlő (/=), 376
nentsel, 232, 235, 273, 275, 303, 462, 476
billentyűzet bevitel, 237
nentselp, 232, 235, 275, 462, 479
billentyűzet bevitel, 237
new_dialog, 305, 309, 313, 324, 480, 625
not, 210, 481
~ (bitenkénti NOT), 198, 377
Boole, 412
növelés (1+), 198, 378
nth, 206, 481
null, 210, 481
numberp, 210, 482
obj_name, 259
obj_path, 260
open, 254, 482
or, 205, 484
Boole, 205, 412
logior, 471
osnap, 239, 303, 484
összeadás (+), 190, 198, 250, 374
osztás (/), 198, 250, 375
parancs, 198, 224, 303
és rajzelemadat kezelő függvények, 279
várakozás felhasználói bevitelre, 225
plusz jel (+), összeadás, 190, 198, 250, 374
polar, 239, 333, 335, 484
prin1, 200, 220, 227, 485
hibakereső információk kiírása, 304
princ, 164, 191, 200, 203, 217, 227, 337, 486
és hibaüzenetek, 217
hibakereső információ kiírása, 304
print, 200, 220, 486
hibakereső információ kiírása, 304
progn, 210, 486
prompt, 200, 202, 303, 487
quit, 216, 487
Lásd még AutoLISP függvények, exit
Lásd még close
quote, 208, 210, 468, 487
read, 318
read-char, 228, 254, 256, 488
read-line, 228, 254, 256, 489
redraw, 231, 303, 489
regapp, 289, 293, 490
relációjelek
egyenlő (=), 375
kisebb vagy egyenlő (<=), 376
kisebb, mint (<), 376
nagyobb vagy egyenlő (>=), 377
nagyobb, mint (>), 377
nem egyenlő (/=), 376
rem, 198, 491
repeat, 205, 491
reverse, 206, 491
rtos, 200, 243, 492
S::STARTUP, 163, 219
set, 210, 493
set_tile, 324, 340, 493
setcfg, 224, 493
setfunhelp, 212, 256, 494
setq, 193, 196, 198, 210, 330, 335, 337, 495
setvar, 224, 226, 337, 495
sin, 198, 242, 496
slide_image, 320, 324, 497
snvalid, 298, 497
sqrt, 198, 497
ssadd, 264, 271, 273, 498
ssdel, 264, 271, 498
ssget, 193, 264, 303, 304, 499, 572
és globális karakter összehasonlítás, 204
kiválasztási opciók, 265
sslength, 264, 272, 504
ssmemb, 264, 272, 504
ssname, 264, 504
start_dialog, 305, 314, 324, 505
start_image, 320, 324, 506
start_list, 317, 324, 506

startapp, 218, 505
strcase, 199, 200, 507
strcat, 199, 200, 507
strlen, 199, 201, 508
subst, 206
substr, 199, 201, 207, 508
szimbólumkezelő függvények, 210
szimbólumtáblák, 298
szorzás (*), 198, 374
szövegpanel, 239
táblák, 257
tablet, 256, 509
tblnext, 288, 298, 510
tblobjname, 298, 512
tblsearch, 288, 293, 298, 512
term_dialog, 317, 513
terpri, 513
textbox, 239, 513
textpage, 303, 514
textscr, 303, 514
törtjel (/), osztás, 250, 375
trace, 210, 514
trans, 243, 251, 516
  disp argumentum, 251
  from argumentum, 251
  pt argumentum, 251
  to argumentum, 251
type, 210
unload_dialog, 324, 518, 625
untrace, 210, 518
vector_image, 320, 324, 519
végrehajtás, 261
ver, 224, 519
vmon, 520, 567
vports, 520
wcmatch, 199, 204, 521
while, 205, 335, 523
write-char, 228, 254, 524
write-line, 228, 254, 524
xdroom, 289, 296, 525
xdsize, 289, 296, 525
xload, 166, 167, 218, 526
xor, Boole, 412
xunload, 167, 218, 527
zerop, 198, 527
Lásd még függvények

B

beágyazott
  párbeszédpanelek, 313
BEDIA parancs, 160
beépített függvények, 423
befejezés
  Lásd még lezárás
BEILL parancs, 433
bejegyzés, alkalmazásnév, 293
belső mezők
  entmake függvény, 282
  entmod függvény, 281
beolvasás
  AutoCAD beviteli eszközök, 455
  EPS, 82
  fájlok, 254
  karakterek, 488
  karakterláncok, 488, 489
BETÖLT parancs, 49
betöltés
  ADS alkalmazások, 166, 526
  ARX alkalmazások, 386
  AutoLISP alkalmazások, 161, 220
  DCL fájlok, 302, 469
  Lásd még memóriából való törlés
betűkonverziók
  DIESEL, 155
bezárás
  párbeszédpanelek, 302
big-fontok, 63
billentyűzet
  beolvasás
    karakter, 488
    karakterláncok, 489
  tetszőleges adatbevitel, 237
billentyűzet-rövidítések, a párbeszédpanelekben, 641
bináris adat, bővített adatban, 292
bitenkénti NOT függvény (~), 198
BLOCK rajzelemtípus, csoportkódok, 591
BLOCKS szimbólumtábla, 288
BLOKK nyomógomb, engedélyezés/letiltás, 378
blokkok
  csoportkódok, 590
  definíciók
    egymásba ágyazás, 431, 477
    létrehozás, 431
    megváltoztatás, 434
  hivatkozások
    attribútumok
      definíciós adatok, 477
      képernyő ábrájának frissítése, 436
      kiválasztás, 477
      létrehozás, 431
      megváltoztatás, 434, 436
    definíciós adatok, 477
    kiválasztás, 477
  létrehozás, 283
  névtelen, 269, 283, 432
bmake.lsp mintaprogram, 314, 325
BODY rajzelemtípus, csoportkódok, 578
Boltzman-állandó (k), 250
bővített adatok
  csoportkódok, 290
  függvények, 289, 367, 490, 525
  kódok, 267
  lekérdezés, 290
  sorszámok, 296
  szervezés, 290
  szűrés, 269
burkolólapok
  rajzolás, 334

C

c (fénysebesség vákumban), 250
C programozási nyelv, és ADS alkalmazások, 166
C:BEÁLL rutin, 258
címkék
  elérése AutoLISP-ből és ADS-ből, 118
CIRCLE rajzelemtípus, csoportkódok, 579
CMDACTIVE rendszerváltozó, 304
CMDDIA rendszerváltozó, 160

CMDECHO rendszerváltozó, 214
csatolások
azonosítók(ASE), 261
elérési útvonalnevek (ASE), 260, 398
függvények (ASE), 366, 395
információ, exportálás, 533
kezelés, 533
kiválasztás (ASE), 261
csendes kilépés, 203
csendes kilépés a függvényből, 217
cserélés. *Lásd* váltás
csillag (*)
mértékegységdefiníciók, 249
névtelen blokknevek, 283, 432
csomóponti területek felszabadítása, 564
csomópontok
definiált, 563, 564
felszabadítás, 564
hulladékgyűjtés, 564, 566, 567
karakterlánc-terület, 565, 566, 567
manuális allokáció, 440, 565, 566
statisztika, 472
szabad, 472, 564
szegmensek, 564, 566
szimbólum tárolása, 565
virtuális függvénylapozás, 520
csonkítás, 442
csoportablakok a párbeszédpanelekben, 643
csoportkódok
blokkok, 590
mezők a bővített adatokhoz, 290
nem-grafikus rajzelemek, 588
szimbólumtáblák, 590
szótár, 589
Lásd még DXF csoportkódok
csoportok, 666
DCL zónák, 629, 666
előre definiált, 642
forgatókönyv mint, 158
vizsgálat, 270
Lásd még DCL zónák
csoportosító operátorok, 270
csúszkák, 312
párbeszédpanelekben, 645, 655, 661, 680
visszajelzési indíték kód, 308
CVPORT rendszerváltozó, 496

D

DCL, 338, 617, 649
akciók, 309
érték, párbeszédpanel attribútum, 443
karakterláncok, 650
megjegyzések, 624
szintaxis, 622
DCL fájlok, 301
betöltés, 302, 340, 469
hierarchia, 619
létrehozás, 339
más DCL fájlokra történő hivatkozás, 620, 621
memóriából való törlés, 518
szemantikai ellenőrzés, 625
szerekezet, 620
DCL fájlok szemantikai ellenőrzése, 625
DCL zónák, 617, 649, 651, 665
a hibák megjelenítésére, 646
aktív (előre definiált), 665
alapértelmezés szerinti értékek, 640
áttekintés, 369
boxed_column, 668
boxed_radio_column, 669
boxed_radio_row, 669
boxed_row, 670
button, 670
column, 671
concatenation, 671
csoportok, 629, 643, 666
csúszka
és a $reason változó, 307
definíciók, 622
dekoratív, 667
dialog, 672
edit_box, 672
és a $reason változó, 307
előre definiált, 642
előre definiált attribútumok, 624, 650, 651
action, 654
alignment, 654
allow_accept, 654
aspect_ratio, 655
big_increment, 655
children_alignment, 655
children_fixed_height, 656
children_fixed_width, 656
color, 656
edit_limit, 657
edit_width, 657
fixed_height, 658
fixed_width, 658
height, 658
initial_focus, 659
is_bold, 659
is_cancel, 659
is_default, 659
is_enabled, 660
is_tab_stop, 660
key, 660
label, 660
layout, 661
list, 661
max_value, 661
min_value, 662
mnemonic, 662
multiple_select, 663
small_increment, 663
tabs, 663
value, 664
width, 664
errtile, 673
felhasználó által definiált attribútumok, 664
gyakorló példa, 340
hivatkozások, 622, 623
idézőjeles karakterláncok, 650
ikon. *Lásd* DCL zónák, image_button
image, 673
image_button, 674
és a $reason változó, 307

kilépő nyomógombok számára, 667
letiltás, 639, 659
list_box, 674
és a $reason változó, 307
nevek, 623
ok_cancel, 675
ok_cancel_help, 675
ok_cancel_help_errtile, 676
ok_cancel_help_info, 676
ok_only, 675
paragraph, 676
popup_list, 677
prototípusok, 619
radio_button, 678
radio_column, 679
radio_row, 679
rádiógomb értékek, 311
részösszeállítások, 619
row, 680
slider, 680
slider_info, 312
spacer, 682
spacer_0, 683
spacer_1, 683
számlap. *Lásd* DCL zónák, csúszka
tájékoztató jellegű, 667
text, 681
text zóna, 667
text_part, 682
tiltott, 668
toggle, 682
definíció törlését elvégző függvények, 564
definíciós adatok
objektumok (rajzelemek), 427, 430, 433, 476
DEFTÖRÖL parancs, 213
diabemutatók, 160
DICTIONARY rajzelemtípus, csoportkódok, 589
DIESEL, 139
állapotsor konfiguráció, 154
AutoLISP kifejezések, 148, 212
használat, 139
hibakereső kifejezések, 148
hibaüzenetek, 149, 155
$(++), 156
$?, 155
idézőjelek, 142
karakterláncok
hosszúság, 155
részkarakterláncok, 155
kifejezések
get *xxx*, 145
menükben, 115
menütételekben, 101
kifejezések parancssorba, 148
makrókifejezések
$ (dollár jel), 141
menükifejezések, 144, 231, 474
DIESEL karakterlánc-függvények
- (kivonás), 149
!= (nem egyenlő), 150
* (szorzás), 149
+ (összeadás), 149
/ (osztás), 149
< (kisebb mint), 150
<= (kisebb egyenlő), 150
= (egyenlő), 149
> (nagyobb mint), 150
>= (nagyobb egyenlő), 150
angtos, 151
csillag (*), szorzás, 149
dátum, 151
edtime, 151
egyenlő (=), 149
eq, 152
és, 150
eval, 153
fix, 153
getenv, 153
getvar, 141, 153
idő, 151
if, 153
index, 153
katalógus, 149
kisebb egyenlő (<=), 150
kisebb mint (<), 150
kivonás (-), 149
konverziók
betű, 155
egész számok, 153
valós számok, 153, 154
linelen, 154
mínusz jel (-), kivonás, 149
nagy, 155
nagyobb egyenlő (>=), 150
nagyobb mint (>), 150
nem egyenlő (!=), 150
nth, 154
or, 154
osztás (/), 149
plusz jel (+), összeadás, 149
rtos, 154
strlen, 155
substr, 155
szögek konvertálása karakterláncokká, 151
szorzás (*), 149
törtjel(/), osztás, 149
XOR, 155
digitalizálók
kalibrációk, 509
menük beállítása, 124
DIMENSION rajzelemtípus, csoportkódok, 579
dimenziók
módosítás, 281
DIMSTYLE rajzelemtípus, csoportkódok, 591
DIMSTYLE szimbólumtábla, 288
DIMZIN rendszerváltozó, 243
DOS
elérési útvonalnév, 255
hivatkozás menükre, 117
menü fájlok, 87
menük vezérlése, 228
parancsok. *Lásd* külső parancsok
rajzok
lásd még grafika
drótváz modellek, renderelés, 557
DSZÖVEG parancs, 198, 417
DXF csoportkódok, 266, 279, 571, 572
grafikus rajzelemek számára, 575
objektumtípus lista, 576
sorszám szerinti lista, 572

E

e (elektron töltése), 250
egész számok
  bővített adatban, 291
  DIESEL, 153
  ellenőrzés, 482
  felhasználói adat bevitele, 449
  felhasználói adatbevitel, 233, 234
  konverziós függvények, 410, 416, 442, 466, 492
  korlátok, 449
  szimbólumok átalakítása, 318
  Lásd még számok
egész típusú számok, 191
egyedi objektumkiválasztás, 101
egyképernyős rendszerek, 514
egymásba ágyazás
  blokkdefiníciók, 431, 477
  függvények, memóriahasználat, 563
  párbeszédpanelek, 640
  rajzelemek, 436
egyszeri aposztróf ('), 208, 212
egyszerű vonaltípusok, 38
"elégtelen karakterlánc-terület" hibaüzenet, 567
"elégtelen node terület" üzenet, 564, 568
elektron nyugalmi tömege (m), 250
elektron töltése (e), 250
elemek
  a listák első elemének kivételével, 414
  listák első eleme, 414
  listák n-edik eleme, 481
  listák utolsó eleme, 467
  listákhoz való hozzáadás, 420
  reverse, 491
  számok a listákban, 468
elfogad billentyű, 641, 654, 675
elhalványított zónák. *Lásd* DCL zónák, letiltás
ellenőrzés
  atomok, 410
  egész számok, 482
  listák, 468
  negatív számok, 475
  nulla kiértékelés, 527
  szimbólumok, 410, 414, 481
  szimbólumtábla-nevek, 497
  valós számok, 482
ELLIPSE rajzelemtípus, csoportkódok, 580
elmozdulás. *Lásd* pontok
előre definiált változók, 198
első elem, listák, 414
élvizsgálat, 559
ENAME szimbólumtípus, 517
ENDBLK rajzelemtípus, csoportkódok, 593
Enter billentyű
  adatbevitel lezárásként, 159
  parancsként, 159
  tilos a felhasználónak válaszként megadni, 463
EPS fájlok, 75
EPS fájlok, rajzok átadása, 75
ERRNO rendszerváltozó, 598
érvényesítés. *Lásd* ellenőrzés
escape karakter (\e), 202
exkluzív or függvények. *Lásd* AutoLISP függvények, xor
exportálás, 531
  3D Studio fájlok, 531
  csatolás információ, 533
EXSUBR szimbólumtípus, 517
external subr függvények, 191

F

fájlformátumok
  EPS, 75
  TIFF, 561
  Unicode fontokhoz, 72
fájlkiterjesztések
  .shp, 50
  .ahp, 23, 29, 461
  .c, 166
  .dcc, 24
  .dcl, 24, 469, 620
  .exe, 166
  .exp, 166
  .hlp, 255, 461
  .lin, 24, 38
  .lsp, 24, 161, 194, 195, 219, 220, 469
  .mln, 24
  .mnd, 24, 87
  .mnl, 24, 87, 162, 163, 168, 169, 219
  .mns, 24
  .mnu, 24, 85
  .pat, 25
  .pcp, 25
  .pfb, 50
  .scr, 25, 158
  .shp, 25
  .shx, 50
  .slb, 321
  .sld, 321
  kilistázott, 697
fájlleírók, 193
fájlok
  3D Studio, 530
  acad.ads, 167
  acad.ahp, 29
  acad.cfg, 20, 445, 493
  acad.dce, 625
  acad.dcl, 621
  acad.hlp, 29
  acad.ini, 24
    AppData szakasz, 227
  acad.lin, 24, 38
  acad.lsp, 143
  acad.lsp, 24, 162, 164, 219, 566
  acad.mnl, 24, 164
  acad.mns, 24
  acad.mnu, 24
  acad.msg, 25, 36
  acad.pat, 25, 44
  acad.pgp, 25, 28
  acad.psf, 25, 75
  acad.unt, 248
  acad.unt, 25
  acadenv.ini, 24
  acadmode.lsp, 143
  acadpsd.ps, 83
  acadpsf.ps, 83
  acadpss.ps, 83
  ai_utils.lsp, 195
  ascii.lsp, 247
  ascii.txt, 247, 689
  base.dcl, 621, 632, 665, 667
  beolvasás, 254
    karakterek, 488
    karakterláncok, 489
  betöltés, 469
    függvények, 219

bmake.dcl, 325
CTRL-Z fájl vége jelek, 483
DCL, 301, 302, 339
ddkj.dcl, 339
ddkj.lsp, 340
dia, 321
ensugom.ahp, 494
fájlkezelő függvények, 228, 254
fontmap.ps, 25, 83
globe.3ds, 530
hello.dcl, 302
írás
karakter, 524
karakterlánc, 524
keresés, 254, 441
kiegészítő, 23
kilistázott, 694
kj.lsp, 328, 330, 333, 336
lezárás, 416
megnyitás, 482
mértékegység-definiálás, 249
.mnl, 164
mode1.lsp, 143
nevek, felhasználói adat bevitele, 447
nil értékre állítás, 565
PostScript, 548
sajatalk.ahp, 495
setup.scr, 159
.shp, 51
.shx, 51
txt.shp, 61
fájlok *morehelp.ahp*, 256
fájltípusok, kilistázott, 697
fejléc fájlok, ADS, 166
felhasználó által definiált függvények, 215
felhasználó által definiált programozási felület, 157
felhasználói adatbevitel
hibaellenőrzés, 235
felhasználói bevitelre szolgáló függvények
adatbevitel
érvényesség, 238
opciók, 235
áttekintés, 357
billentyűzet bevitel, 237
bits argumentum, 236
feltételek vezérlése, 235
Lásd még egyedülálló, 357
prompt argumentum, 235
string argumentum, 236
Felhasználói koordináta-rendszer. *Lásd* FKR
felhasználói megjegyzések, 251
felhasználók figyelmeztetése hiba esetén, 646
felkiáltójel (!), 332
és változók, 197
feltételes függvények, áttekintés, 349
felületenkénti csomópontok, maximum, 432
FÉNY parancs, 534
függvény módok, 534
fények
átnevezés, 540
létező módosítása, 536
létező törlése, 535
listázás, 535
szórt háttérvilágítás, 535
új
pont, 538
reflektorok, 539
távoli, 537
fénysebesség vákumban (c), 250
figyelmeztető ablakok a párbeszédpanelekben, 646
FILE szimbólumtípus, 517, 518
fájlok
lásd még DCL fájlok
FKÖNYV parancs, 158, 417
FKR, 242, 251, 275, 484, 515
fokok, átalakítás radiánba, 246, 328
FÓLIA nyomógomb, engedélyezés/letiltás, 378
fólianevek, bővített adatban, 291
fontfájlok, 50
fontok
Adobe, 83
big-fontok, 63
PostScript, 50
saját létrehozása, 61
speciális szimbólumok, 73
szövegfontok leírása, 61
Unicode, 71
fonttábla-fájl (*fontmap.ps*), 83
fordítás, PostScript fontok, 50
FORGAT3D parancs, 540
forgatókönyv fájlok
létrehozás AutoCAD-en kívül, 158
szóközök, 159
forgatókönyv szolgáltatás, 158
Freeplot jellemző, 159
FRISSÍT parancs, 231
frissítés, objektum képernyő ábra, 436
függvények
ADS alkalmazások, 381, 526
alkalmazáskezelés, 354, 416, 490
argumentumokkal, 215
aritmetikai, 346
ARX alkalmazások, 385
ASE felület, 362
áttekintés, 345, 366
AutoLISP. *Lásd* AutoLISP függvények
beépített, 423
betöltés fájlokból, 219
billentyűk, Képváltó, 454, 514
bővített adatok kezelése, 289, 367, 525
definiálás, 421, 494
definíció törlése, 564
DIESEL, 149
egyenlőség, 349
egymásba ágyazás, 563
eszközelérés, 256, 361
eszközvezérlés, 256
fájlkezelés, 228, 254, 361
felhasználói adatbevitel, 357
feltételes, 349
függvénykezelés, 353
geometriai, 359
hibakezelés, 235, 353, 392
karakterlánc-kezelés, 348, 358, 360
karakterlánc-konverzió, 360
képernyővezérlés, 227
kezelés, 210
kiválasztási halmaz, 192, 263, 498
konverziók, 243
Lásd még konverziós függvények
kulcsszavak, 462
külsőleg definiált, 373

lapozási táblák, 568
lekérdezés, 224
listázás, 410, 565
lokális szimbólumok, 214
megjelenítés vezérlése, 356
memóriahasználat, 563, 564
mint AutoCAD parancsok, 211
névtelen, 467
objektumkezelés, 272
parancs, 224, 355
programozható párbeszédpanel, 340, 369
rajzelem
elérés, 278
név, 273
rekurzió, 470, 563
szám argumentumok, 373
szimbólumkezelés, 352
szintaxis, 194
visszaadott listák, 471
Windows alkalmazás, 505
zóna és attributum kezelés, 304

G

geometriai függvények, áttekintés, 359
geometriai számológép, 545
GIF képek, 541
megjelenítés, 547
renderelt képek elmentése, 546
GIFBE parancs, 541
a kép szélesség-magasság arányának megváltoztatása, 558
élvizsgálat, 559
használt színek száma, 559
háttérszín meghatározása, 558
képek importálása
szürkeárnyalat, 559
világosságon alapuló képek, 559
globális karakterek
karakterlánc-minta összehasonlítás, 204, 268, 521
szűrőlistákban, 268
globális karakterek listája
- (kötőjel), 521, 523
(font), 521
* (csillag), 204, 269, 521
, (vessző), 521
. (pont), 521
... (három pont), 521
? (kérdőjel), 204, 521
@ (at), 521
~ (hullám), 521
' (fordított aposztróf), 521, 522
görbe vonalláncok, 286
grafika
ablakok, vezérlés, 231
alacsonyszintű, 232
burkolólapok rajzolása, 334
frissítés
nézetablak, 489
objektum, 489
grafikus ablakok vezérlése, 231
grafikus képernyő megjelenítése, 454
képernyő, és adatokat kezelő függvények, 285
vektorok rajzolása
a párbeszédpanel-ábrába, 519
grafikus képernyőn, 454, 459
grafikus ábra, megjelenítés párbeszédpanelben, 674
grafikus képernyő, 232
grafikus kiválasztási halmazok, 534
grafikus rajzelemek, csoportkódok, 575
GRAFKÉP parancs, 454
Grayscale kitöltési minta, 550
GUI platformok, hatása a párbeszédpanelekre, 615, 634
gumivonal/keret, szaggatott vonal, 464
gyakorló példa, AutoLISP, 327
gyorsító billentyűk, 135, 641

H

h (Planck-állandó), 250
halom
csomóponti terület, 564
definiált, 563
karakterlánc-terület, 565, 566, 567
Lásd még csomópontok
határoló keret, 240
háttérszínek, meghatározás, 558
Hiba leíró-kód (ASE), 262
hibadiagnosztika (ASE), 262
hibák, AutoLISP, 163
hibakeresés
DIESEL kifejezések, 148
trace/untrace, 514, 518
hibakezelés, DCL, 624
hibakezelő függvények, 392
áttekintés, 353
hibaüzenetek
"'alkalmazás' alkalmazás már be van töltve", 526
"Az AutoCAD visszautasította a függvényt", 304, 417
"elégtelen karakterlánc-terület", 567
"elégtelen node terület", 564, 568
ASE, 262
DIESEL, 155
felhasználó által definiált, 438
formátum, 216
quit/exit megszakítás, 487
hierarchikus menük. *Lásd* lépcsőzetes menük
hieroglifa fájl. *Lásd* DCL zónák, image_button
hőmérséklet átalakítók, 250
homogén koordináták, 258
honosított változatok, parancsnevek, 417
hosszúság
állapotsor, 140, 154
karakterláncok, 508
korlát, 453
kiválasztási halmazok, 504
listák, 468, 525
szimbólumnevek, 566

HSRAFFOZ parancs
kiegészítés új sraffozási mintákkal, 44
hulladékgyűjtés, 443, 564, 566, 567

I

idézőjelek ("), karakterláncokban, 192, 202, 246
igazítás, AutoLISP programfájlokban, 196
ILLESZT parancs, 532
importálás, 541
3D Studio fájlok, 530
GIF képek, 541
PCX formátumú raszteres képfájlok, 548
PostScript fájlok, 548
szürkeárnyalatú képek, 559
TIFF fájlok, 561
világosságon alapuló képek, 560
include direktíva, 620, 621
adatbevitel
lásd még read
INSERT rajzelemtípus, csoportkódok, 580
INT szimbólumtípus, 517, 518
írás
AppData szakasz (*acad.cfg*), 493
karakterek, 524
karakterláncok, 524
szöveg
állapots, 458
képernyőmenü-területre, 458

J

JELENET parancs, 542
jelenetek, 542
aktuális beállítása, 542, 545
átnevezés, 542
definíciók, 542
létező módosítása, 542
létező törlése, 542
összes kilistázása, 542
új létrehozása, 542
Jellemző hibakód (ASE), 262
Jellemző hibaüzenet (ASE), 262
jelölések, 336

K

k (Boltzman-állandó), 250
KAL parancs, 534
kalibrációk
digitalizálók, 509
táblák, 257
kapcsoló. *Lásd* váltás
kapcsolók a párbeszédpanelekben, 645
kapcsolt értékpárok, 209, 420
karakterlánc konstans, 192
karakterlánc-kezelés, 199, 350
karakterláncok
a párbeszédpanel-listákban, 380
beolvasás, 488, 489
bővített adatokban, 291
felhasználói adat bevitele, 452
felhasználói adatbevitel, 233, 234
globális karakterek, 521
globális karakter-minta összehasonlítás, 204
hosszúság, 453, 508
idézőjelek, 192, 202, 246
írás, 524
konverziós függvények, 360, 507
listák rendezése, 379
literal, 192
memóriahasználat, 565
nagy- illetve kisbetűre történő konvertálás, 507
összefűzés, 507
részkarakterláncok, 508
vesszővel elválasztott, 153
vezérlőkarakterek, 201
visszahajló törtjelek (\), 192, 201, 202, 245, 311, 418, 453
karaktermagasság egysége, 658
karaktertáblázat, 71
kép szélesség-magasság arányok, 558
képernyő
egyképernyős rendszerek, 514
grafikusról szövegesre váltás, 514
írás
karakter, 524
karakterlánc, 524
szöveg, 458
keresztek, 336
megjelenítés
grafikus képernyő, 454
prompt, 487
képernyő koordináta-rendszer (KKR), 252
képernyőalmenük, 95
képernyőmeghajtók, menük módosítása, 89
képernyőmenük, 106
címkézett almenük, 107
tételcímkék, 109
képernyőn
szöveges képernyő törlése, 514
KÉPMENT parancs, 546
függvény argumentumok, 546
KÉPÚJRA parancs, függvény argumentumok, 547
Képváltó billentyű, 454, 514
keresés
asszociációs listák, 408
fájlok, 254
szimbólumtáblák, 512
keresztek a képernyőn, 336
kerti járda parancs, 328
Két-pi kulcsszó, 237
kiegészítő fájlok, 23
kiegészítő fájlok elérési útvonala, 19
kiegészítő menük, 103
kifejezések
kiértékelés, 439, 442, 469, 486
ismétlődve, 491, 523
összefűzés, 468
Lásd még listák
kifejezések kiértékelése, 439, 442, 469, 486
ismétlődve, 491, 523
kihúzási vektorok, 253
kiindulópont, definiált, 330
KIJELÖL parancs, 418

kijelölési pontok, elhelyezése parancsokhoz, 226
KILÉP parancs, helyettesítés, 219
kimenet. *Lásd* írás
kisbetű. *Lásd* nagy- és kisbetűre vonatkozó konvenciók
kiválasztási halmaz szűrőlisták
csoportosító operátorok
, 503
<AND, 503
<NOT, 503
<XOR, 503
relációjelek
!=, 501, 502
& (bitenkénti AND), 501
&= (bitenkénti maszkolt egyenlőség), 501
* ("bármi lehet"), 501, 502
/= (nem egyenlő), 501
< (kisebb, mint), 501
<= (kisebb vagy egyenlő), 501
<>, 501
= (egyenlő), 501, 502
> (nagyobb, mint), 502
>= (nagyobb vagy egyenlő), 501
"bármi lehet" (*), 502
egyenlő (=), 502
kisebb vagy egyenlő (<=), 502
kisebb, mint (<), 502
nagyobb vagy egyenlő (>=), 502
nagyobb, mint (>), 502
nem egyenlő (/=), 502
kiválasztási halmazok, 192, 263
függvények, 263, 402, 498
létrehozás, 264, 534
műveletek, 271
szűrőlisták, 267
törlés a memóriából, 266
KJ parancs, 328
KKR (képernyő koordináta-rendszer), 252
konfiguráció
könyvtárak, 20
vezérlés, 227
konstansok, 192
kontextus-karakterlánc, 30
konverziós függvények
áttekintés, 347, 348, 360
karakterláncok, 243
könyvtár fájlok, ADS, 166
könyvtárak, 288
ads, 166
elválasztójelek
/ (törtjel), 255, 469
\ (visszahajló törtjel), 162, 255, 469, 483
keresés, 18
kofiguráció, 20
objektumok, 288
prototípus-rajz, 22
rajz, 21
sample, 161
support, 161
szerkezet, 19
többféle konfigurációhoz, 20
törtjel, mint elválasztójel, 469
koordináta-rendszerek transzformációja, 251, 477, 479, 515
koordinátatranszformációs adatok, 275
köralakú burkolólapok, rajzolás, 334
korlátok, 140
állapotsor hosszúság, 154
DIESEL függvényparaméterek, 149
egész számok, 449
felületenkénti csomópontok, 432
karakterlánc-hossz, 453
rendszerváltozó értéke, 496
Lásd még AutoLISP függvények, max
Lásd még AutoLISP függvények, min
környezeti változók, 226
ACADCFG, 21
visszakeresés, 446
DIESEL, 153
körvonalak, rajzolás, 334
kulcsérték (ASE), 261
kulcsszavak
felhasználói adatbevitel, 233, 450
létrehozása felhasználói bevitelre szolgáló függvények számára, 462
külső adatbázis
adatok, 533
csatolások kezelése, 533
parancsok, adminisztratív függvények, 532
külső fájlok, AutoLISP kód betöltése, 160
külső parancsok
álnevek, 28
definiálás
AutoCAD parancsok, 213
definiálása, 26
külsőleg definiált
függvények, 373
ADS alkalmazások mint, 165
parancsok
áttekintés, 529
elérés, 529
kurzormenük, 110
címkék, 115
elérése AutoLISP-ből és ADS-ből, 118
megjelölés, 116
tagolás, 115
tiltás, 115
hivatkozás almenükre, 117
lépcsőzetes, 114
maximális tételek, 110
szintaxis, 110
tiltott parancsok, 119

L

lapozási táblák, 568
LAYER rajzelemtípus
entmod függvény és, 433
LAYER rajzelemtípus, csoportkódok, 593
LAYER szimbólumtábla, 288
LEADER rajzelemtípus, csoportkódok, 581
legördülő listák, 317
lekérdezőfüggvények, 355
lépcsőzetes menük, 110

lezárás
fájlok, 416
párbeszédpanelek, 426, 513, 518
tételek, 95
LINE rajzelemtípus, csoportkódok, 581
LINESTYLE szimbólumtábla, 288
LISP programozási nyelv. *Lásd* AutoLISP
LIST szimbólumtípus, 517, 518
lista függvény maradéka, 473
listaablakok
visszajelzési indíték kód, 308
listák, 192
asszociáció, 209, 408
beolvasás, 488
ellenőrzés, 468
első elem, 414
első elem kivételével, 414
építés/összefűzés, 385, 420, 468
fordított elemek, 491
függvény áttekintés, 350
függvényekből visszaadott lista, 471
hosszúság, 468, 525
kapcsolt értékpárok, 209, 420
keresés, 408, 473
kezelés, 206, 350
kifejezések kiértékelése az összes tagra, 442
létrehozás, 317
maradék, 473
n-edik elem, 481
pontlisták, 207
rendezés, 379
tételek hozzáfűzése, 317
tételek módosítása, 317
utolsó elem, 467
listakezelés, 206
listázás
fények, 535
függvények, 410, 565
szimbólumok, 410, 565
literal karakterláncok, 192
LOCAL függvény, 215
logikai csoportosító operátorok, 270
logikai operátorok. *Lásd* kiválasztási halmaz szűrőlisták, relációjelek
lokális szimbólumok
függvényekben, 214
long számok
bővített adatban, 291
LTYPE rajzelemtípus, csoportkódok, 594
LUNITS rendszerváltozó, 243
LUPREC rendszerváltozó, 243

M

m (elektron nyugalmi tömege), 250
MACROTRACE rendszerváltozó, 148
makrók
menü, 85
menü, DIESEL kifejezések, 144
MODEMACRO rendszerváltozó, 140
manuális memória allokáció, 440, 565, 566
mátrix
transzformáció, 254, 258, 477, 479
maximumok. *Lásd* korlátok
megdöntött szövegfontok, 61
megjegyzések
AutoLISP, 161, 195, 328
elválasztó jelek, ; (pontosvessző), 158, 161, 251, 328
elválasztó jelek, / (normál törtjel), 195
felhasználói, 251
forgatókönyv fájlokban, 158
soron belüli, 195
megjelenítés szabályozása, 115
megjelenítésvezérlő függvények
áttekintés, 356
Lásd még egyedülálló parancsok
megnyitás
fájlok, 482
párbeszédpanelek, 469, 480, 505
megnyújtott szövegfontok, 61
Mégsem billentyű, 641
megtalál. *Lásd* keresés
memóriából való törlés
ADS alkalmazás, 527
ARX alkalmazások, 386
DCL fájlok, 518
Lásd még betöltés
memóriakezelés
áttekintés, 563
csomóponti terület, 563, 564
függvény áttekintés, 371
halom, 563, 564
hulladékgyűjtés, 443, 564, 566, 567
karakterlánc-terület, 565, 566, 567
manuális allokáció, 440, 565, 566
statisztika, 472
szegmensek, 564, 566
szimbólum tárolása, 563, 564
verem, 563
virtuális függvénylapozás, 520
menü
fájlok, 86
betöltés, 87
definíció, 87
LISP, 87
menütételek szintaxisa, 94
szakaszok, 90
szerkezet, 90
típusok, 86
makrók, 85
tételek megjelölése
felkiáltójel, 117
pipa, 117
MENÜ parancs, 87
MENÜBETÖLT Menüadaptációs párbeszédpanel, 129
MENÜBETÖLT parancs, 129
MENUCTL rendszerváltozók, 95
MENUECHO rendszerváltozó, 98
menüfájlok
almenük, 93
menük, 110
ábrazóna, 119
adaptálás, 85
adatbevitel engedélyezése, 97
alap, 89

címkék
megjelölés, 116
tagolás, 115
tiltás, 115
címkézés, 95
csere, 119
DIESEL kifejezések, 144, 474
DIESEL kifejezések menükben, 146
használat AutoLISP-ben, 126
hivatkozás, 117
hivatkozás DOS menükre, 117
hivatkozás Windows menükre, 127
honosított nyelv támogatása, 98
kurzor, 110
Lásd még kurzormenük
Lásd még lépcsőzetes menük
legördülő, 110
lépcsőzetes, 110, 114
letiltott echók és promptok, 98
maximális tételek, 110
paraméterek, 94
parancs megszakítása, 98
parancsok ismétlése, 100
rendszeregér használatával, 103
részleges, 87
speciális karakterek, 102
szintaxis, 94, 110
szöveg írása a képernyőmenü területére, 458
tábla, 124
tételek, 114
tételek állapota, 229, 473
tételek megjelölése, 116
tiltott parancsok, 119
vezérlés, 228
vezérlőkarakterek, 99
Windows működése, 86
Lásd még AutoLISP függvények, menucmd
MENÜTÖRÖL parancs, 130
Méret prompt
Átmérő alparancs, 418
Sugár alparancs, 418
méretezések
stílus felülírás, 588
mértékegység neve, 249
mértékegység-definiálás, 249
mértékegységek átalakítása, 248
mértékegységek, konvertálás, 420
minimumok. *Lásd* korlátok
minta összehasonlítás, karakterláncok, 521
mintafájl, 44
mínusz jel (-), 29
MKR (Modell koordináta-rendszer), 275, 477
MLINE rajzelemtípus, csoportkódok, 581
.mnl fájlok, 164
mód, állapotsor. *Lásd* állapotsor
Modell koordináta-rendszer (MKR), 275, 477
Modelltér, 252
modelltér, átkapcsolás a papír és tér között, 144
Modell-Világ transzformációs mátrix, 275, 477
MODEMACRO hosszúság, 140
MODEMACRO rendszerváltozó, 140, 154
MTÉR parancs, 285
MTEXT rajzelemtípus, csoportkódok, 582

N

nagy- és kisbetűre vonatkozó konvenciók
kisbetűre történő konvertálás, 507
kulcsszavak, 465
nagybetűre történő konvertálás, 507
nagybetű
DIESEL, 155
Lásd még, 507
nagybetűs írásmód a párbeszédpanelekben, 638
negatív számok
ellenőrzés, 475
tilos a bevitele, 463
négyszögek
a párbeszédpanelek ábrazónáiban, 440
sarokpont bevitel, 445
nem-grafikus rajzelemek, csoportkódok, 588
nevek
alkalmazásnevek bővített adatban, 291
ASE csatolás-elérési útvonalak, 399, 405
ASE xrefek vagy blokkok, 407
átnevezés
fények, 540
jelenetek, 542
DCL zónák, 623
fájlok, 447
függvény, 211
névtelen blokkok, 283, 432
objektumok, 429, 434, 460, 504
rajzelem, 193, 264
NÉVJEGY parancs, 36
névtelen
blokkok, 283
*U*nnn*, 432
létrehozás, 432
szűrés, 269
függvények, definiálás, 467
nézetablak
entmake függvény, 282
entmod függvény, 281
frissítés, 489
leírók, 520
törlés, 454
vektorok rajzolása, 454, 459
nil
és csendes kilépés, 203
és globális karakter összehasonlítás, 204
hozzárendelés változókhoz, 197
tétel ellenőrzése
kiértékelés eredménye nil, 481
nil érték tartozik hozzá, 481
nulla
szám kiértékelésének ellenőrzése, 527
tilos a bevitele, 463
nyelvek
párbeszédpanelekben történő használat, 641

nyitás
párbeszédpanelek, 302
nyomógomb-menük, 103
nyomógombok, 319, 642

O

objektum (rajzelemek)
frissítés, 489
Objektum koordináta-rendszer. *Lásd* OKR
objektumkezelés, 366
objektumkezelő függvények, 272
objektumok (rajzelemek)
ASE adatbázis, 388, 402
bővített objektum adatok, függvények, 367, 490, 525
definíciós adatok, 427, 430, 433, 476
DXF csoportkódok, 571
egymásba ágyazás, 436
képernyő ábrájának frissítése, 436
kiválasztás, 435, 476, 499
kiválasztási halmazok, 402, 498
létrehozás, 430
nevek, 429, 434, 460, 504
összetett, 431, 476
törlés, 427, 498
visszaállítás, 427
objektumok forgatása, 540, 541
objektumok igazítása, 541
objektumok kiválasztása, 366, 435, 476, 499
objektumok mozgatása, 541
objektumok tükrözése, 542, 561
OK nyomógomb, zónaértékek lekérdezése, 305
OKR, 252, 515
kezdőpont, 241
oktális kódú karakter (\nnn), 202, 203
OLEFRAME rajzelemtípus, csoportkódok, 582
on-line geometriai számológép, 545
on-line súgó. *Lásd* súgó
orientáció. *Lásd* AutoLISP függvények, getorient
ortogonális transzformáció, 259
összefűzés
car és cdr függvények, 415
kifejezések, 468
listák, 385, 420, 468
összegzés
Lásd AutoLISP függvények, összeadás
Lásd DIESEL karakterlánc-függvények, összeadás (+)
összenyomott szövegfontok, 61
összetett objektumok (rajzelemek), 431, 476
feldolgozás, 283
létrehozás, 283
összetett vonaltípusok, 38, 40
alakok, 41
szöveg, 42
oszlopnevek (ASE), 260

P

PAGETB szimbólumtípus, 517
papírtér képernyő koordináta-rendszer (PTKKR), 252
papírtér, átkapcsolás papír és tér között, 144
parancs
automatikus betöltés (AutoLISP), 164
GIFBE, 559
PCXBE, 559
TIFFBE, 559
parancs függvények, AutoCAD parancsok elérése, 529
Parancs prompt
szöveg beírása, 190
parancs-forgatókönyvek, 158
parancsírási lehetőség, 157
parancsnév előtagok, . (pont), 213
parancsok
3DSBE, 530
3DSKI, 531
acad, 158
ALAK, 49
ALKTÖLT, 161
álnevek, 28
ANYAGTÁR, 532
ASE, 169
ASEADMIN, 532
ASEEXPORT, 533
ASELINKS, 533
ASEROWS, 533
ASESELECT, 534
ASESQLED, 534
ASEUNLOAD, 169
átdefiniálás, 211, 213
ATTDEF, 198
ATTEDIT, 279
BEDIA, 160
BEILL, 433
BETÖLT, 49
DEFTÖRÖL, 213
DSZÖVEG, 198
fejlesztés, 327
FÉNY, 534
FKÖNYV, 158
FORGAT3D, 540
FRISSÍT, 231
függvény áttekintés, 355
függvények mint, 211
GIFBE, 541, 558, 559, 560
honosított változatok, 417
ILLESZT, 532
JELENET, 542
KAL, 534
KÉPMENT, 546
KÉPÚJRA, 547
keresési eljárás, 22
kijelölési pontok elhelyezése, 226
KILÉP, helyettesítés, 219
külsőleg definiált, 529
MENÜ, 87
MENÜBETÖLT, 129
MENÜTÖRÖL, 130
MTÉR, 285
NÉVJEGY, 36
PCXBE, 548, 558, 559, 560
PLOT, 160
pont előtag, 225
PSBE, 82, 550
PSKÉP, 548
PSKI, 75
PSKITÖLT, 549
PTÉR, 285
RANY, 550
RBEÁLL, 554
REGEN, 286
RÉGIDEF, 213
RENDER (AutoCAD), 169

RENDER (külsőleg definiált), 557
RENDERTÖRÖL, 169
RENDKÉP, 557
RKONFIG, 550
RSTAT, 560
SHELL, 26
SOKLAP, 432
STILUS, 61
SÚGÓ, 29
súgó a felhasználó által definiált parancsokhoz, 494
SZÖVEG, 198
TÁBLA, 124, 257
TIFFBE, 558, 559, 560, 561
TÜKRÖZ3D, 542
ÚJ, 22
új hozzáadása, 211, 212, 336
várakozás felhasználói bevitelre, 225
VÉGE, helyettesítés, 219
végrehajtás, 416
VISLKONV, 562
VISSZA, 158
VLÁNC, 213
VLEDIT, 279, 286
VONAL, 198
VTIPUS, 38
parancsok végrehajtása, 416
parancssor
DIESEL kifejezések, 148
függvények, 211
írás
újsor, 513
üzenetek megjelenítése, 227
változók használata, 197
párbeszédpanelek, 301, 338
ábrazónák
függvény áttekintése, 370
Lásd még AutoLISP függvények, menucmd
ADS és, 157
alapértelmezés
elrendezés, 629
értékek, 640
alkalmazás-specifikus adat, 323
alkotó elemek. *Lásd* zónák
attribútumok, 443
AutoLISP és, 157
bezárás, 302
cél, 338
egymásba ágyazás, 313, 640
elrejtés, 304, 314, 325, 640
függvények összefoglalása, 324
gyakorló példa, 338
hibakezelés, 636, 646
hozzáadás, 480
inicializálás, 480
kiegészítés, 338
kilépő nyomógomb szövege, 632
legördülő listák, függvény áttekintés, 370
lezárás, 317, 426, 513, 518
listaablakok, függvény áttekintés, 370
más nyelvekre fordítás, 641
megnyitás, 302, 469, 480, 505
nyomógombok, 642
programozható párbeszédpanel függvények, 340
áttekintés, 369
súgó, 637
szerkesztőablakok, 643
tervezés, 634
GUI felület, 634
hátrányos helyzetű felhasználók számára, 637
nagybetűs írásmód, 638
nyelv, 635
nyomógomb, 635
többszörös, 636
új definiálása, 672
visszajelzések, 307
zónák, 309
asszociációs alkalmazás-kezelt adatok, 416
függvény áttekintés, 309, 369, 371
gyakorló példa függvény, 340
zónák közti tér beállítása, 630
Lásd még DCL zónák
párbeszédpanelek elrejtése, 314, 325
párbeszédpanelek inicializálása, 480
Párbeszédvezérlő nyelv. *Lásd* DCL
PAUSE előre definiált szimbólum, 198, 225
PAUSE szimbólum, 418
PCX formátumú raszteres képfájlok importálás, 548
PCXBE parancs, 548
a kép szélesség-magasság arányának megváltoztatása, 558
élvizsgálat, 559
használt színek száma, 559
háttérszín meghatározása, 558
imp, 559
importálás világosságon alapuló képek, 560
PDB szolgáltatás. *Lásd* DCL zónák
PFACEVMAX rendszerváltozó, 432
pi kulcsszó, 237
PICKADD rendszerváltozó, 158, 161, 165
PICKAUTO rendszerváltozó, 158, 161, 165
PICKSET szimbólumtípus, 517
Planck-állandó (h), 250
PLOT parancs, 160
"plusz vagy mínusz" szimbólum, 204
POINT rajzelemtípus, csoportkódok, 582
POLYLINE rajzelemtípus, csoportkódok, 583
pont előtag parancsokban, 225
pontlisták, 207
pontok
Lásd még AutoLISP függvények, vektorok
3D, 484
beszúrás, 240
felhasználói adat bevitele, 451, 463
felhasználói adatbevitel, 233, 234, 331
kiindulópont, 330

koordináta-rendszerek
transzformációja, 477, 479, 515
változó definíció, 468
Y koordináta, 254, 415, 467
Z koordináta, 254, 415, 467
Lásd még AutoLISP függvények, distance
pontszerű fényforrások, létrehozás, 538
ponttranszformációk, 254
PostScript
ábrák
megjelenés vezérlése, 549
EPS fájlok átadása, 75
értelmező program (*acadps*), 82
fájlok
fóliaváltások, 81
importálás, 548
színváltások, 82
vonaltípus-váltások, 82
font
helyettesítő tábla, 76
fordító fontok, 50
képek beolvasása, 82
kitöltési minta, vonallánc
körvonalak kitöltése, 549
saját kitöltési minták definiálása, 79
Postscript
font
fájl, 50
hivatkozási anyag, 84
ISO 8859 Latin/1 szerinti újrakódolása, 77
kétdimenziós vonalláncok kitöltése, 75, 77
pozícionáló eszköz, adaptálási menü, 103
pretty printerek, és LISP kód, 196
programozható párbeszédpanelek, 617
AutoLISP függvények
dimx_tile, 306
dimy_tile, 306
programozható párbeszédpanel függvények, 617
programozható párbeszédpanelek (PDB). *Lásd* DCL zónák
programparaméter-fájl, 25
projektív transzformáció, 259
prototípus-rajzok, 22
PSBE parancs, 82, 550
PSKÉP parancs, 548
PSKI parancs, 75
PSKITÖLT parancs, 549
PSPROLOG rendszerváltozó, 81
PSQUALITY rendszerváltozó, 82
PTÉR parancs, 285
PTKKR (papírtér képernyő koordináta-rendszer), 252

Q

quit/exit megszakítás, 439

R

r1 szorzótényező, 250
r2 hozzáadott eltolás, 250
radián
fokok átalakítása, 328
szögek, 151, 328
lásd még szögek
rádiócsoportok, 311
rádiógombok, -sorok és -oszlopok, 644
rajzelem
adatokat kezelő függvények, 279
és a grafikus kérernyő, 285
definíciós listák, 240, 264
elérési függvények, 278
nevek, 193, 263, 264
névfüggvények, 273
összefüggés, 275
sorszámok, 274
szűrőlisták, 266
rajzelemek. *Lásd* objektumok (rajzelemek)
Rajzfájlok kiválasztása párbeszédpanel, 19
rajzok, 82
EPS fájlok átadása, 75
prototípus, 22
RANY parancs, 550
"N" (NEW) mód argumentumok, 553
függvény módok, 550
RAY rajzelemtípus, csoportkódok, 584
RBEÁLL parancs, 554
REAL szimbólumtípus, 517, 518
reflektorok, létrehozás, 539
REGEN parancs, 286, 436
RÉGIDEF parancs, 213
REGION rajzelemtípus, csoportkódok, 584
rekurzió
függvény betöltése, 470
memóriahasználat, 563
relációvizsgálat, kiválasztási halmazokban, 269
RENDER parancs, 557
(AutoCAD), 169
renderelés, 550
anyagok, 551, 552
definíciók, 552
beállítás újrakonfigurálása, 560
beállítások, 554
képfájl formátumok, 546
kiválasztás
kimeneti eszköz, 555
paletta opció, 554
meghatározás
ikonblokk méret, 555
render típus, 555
objektum kiválasztásra történő felszólítás, 556
opciók, 556
statisztika, 560
Visual Link adatok, 562
RENDERTÖRÖL parancs, 169
RENDKÉP parancs, 557
rendszeregér, 103
rendszerkönyvtár, 19
rendszerváltozók, 226, 529
ANGBASE, 234, 244, 496
ANGDIR, 234
APERTURE, 239
AUNITS, 243
AUPREC, 243
CMDACTIVE, 304
CMDDIA, 160
CMDECHO, 214
CVPORT, 496

DIMZIN, 243
ERRNO, 598
érték beállítása, 495
LUNITS, 243
LUPREC, 243
MACROTRACE, 148
MENUCTL, 95
MENUECHO, 98
MODEMACRO, 140
PFACEVMAX, 432
PICKADD, 158, 161, 165
PICKAUTO, 158, 161, 165
PSPROLOG, 81
PSQUALITY, 82
RIASPECT, 558
RIBACKG, 558
RIGAMUT, 559
SCREENBOXES, 109
SNAPANG, 496
SORTENTS, 239
SPLINESEGS, 287
TABMODE, 257
TEXTEVAL, 198
UNDOCTL, 217
UNITMODE, 245
visszakeresés, 141, 453
DIESEL, 153
visszatérés, 337
Lásd még környezeti változók
részleges menük, 87, 129
részösszeállítások, 666
RIASPECT rendszerváltozó, 558
RIBACKG redszerváltozó, 558
RIGAMUT redszerváltozó, 559
RKONFIG parancs, 550
rövidítések
a párbeszédpanelekben, 639
elkerülés, 159
parancsok. *Lásd* parancsok, álnevek
RSTAT parancs, 560

S

sample könyvtár, 161
sarkok, felhasználói adat bevitele, 445
sarkok, felhasználói adatbevitel, 233, 234
SCREENBOXES rendszerváltozók, 109
segédfüggvények, 355
SEQEND rajzelemtípus, csoportkódok, 584
sétáló menük. *Lásd* lépcsőzetes menük
SHAPE rajzelemtípus, csoportkódok, 584
SHELL parancs, 26
.shp fájlok, 51
.shx fájlok, 51
síkról visszaverődő objektumok, 561
SKICC parancs, 417
SNAPANG rendszerváltozó, 496
SOKLAP parancs, 432
SOLID rajzelemtípus, csoportkódok, 584
soron belüli megjegyzések, 195
sorszámok, 263, 274
bővített adatban, 296
SORTENTS rendszerváltozó, 239
speciális kódok
Lásd alakdefiníciós fájlok, speciális kódok
specifikus fájlnevek, 361
SPLINE rajzelemtípus, csoportkódok, 585
spline vonalláncok, 286
SPLINESEGS rendszerváltozó, 287
SQL utasítások, végrehajtás, 534
sraffozási minták, 45
készítés, 44
módosítás, 281
STILUS parancs, 61
STR szimbólumtípus, 517, 518
STYLE szimbólumtábla, 288
SUBR szimbólumtípus, 517, 518
Sugár alparancs, 418
súgó
a felhasználó által definiált parancsokhoz, 494
fájlformátum, 30
meghívás, 379, 461
menü-specifikus, 134
súgó direktívák, 30
SÚGÓ parancs, 29
súgófájl megjelenítő, 29
hiperszöveg csatolások, 33
súgószöveg formázása, 32
súgófájlok
adaptálás, 29
R12 súgófájlok konvertálása R13 súgófájlokká, 34
súgómegjelenítő, 31
support könyvtár, 161
SYLE rajzelemtípus, csoportkódok, 594
SYM szimbólumtípus, 517, 518
szabad
ASE csatolás kiválasztás, 403
csomópontok, 472, 564
szaggatott vonalak, gumivonalak vagy -keretek számára, 464
számlap zóna. *Lásd* DCL zónák, csúszka, 649
számok
konverziós függvények, 347, 360
mint függvények argumentumai, 373
negatív
ellenőrzés, 475
negatív számok
tilos a bevitele, 463
nulla kiértékelés, ellenőrzés, 527
Lásd még egész számok
szegmensek, 564, 566
szerkesztőablakok, 313, 643, 672
visszajelzési indíték kód, 308
szerkesztőprogramok, ADS alkalmazások kifejlesztésére, 166
szimbólumok, 193
beolvasás, 488
definíció törlése, 564
ellenőrzés, 410, 414, 481
értékbeállítás, 330, 337, 493
kezelés, 210
kifejezés kiértékelés nélküli visszaadása, 487
listázás, 410, 565
lokális, 214
memóriahasználat, 563, 564
mint argumentumok, 215
név hossza, 566
PAUSE, 198, 225, 418
típusok, 517
szimbólumok, egész számmá alakítás, 318
szimbólumtáblák, 263

csoportkódok, 590
szimbólumtábla-kezelés, 368
színek
háttér, 558
száma, 559
Színválasztó párbeszédpanel, 320, 378
szögek
felhasználói adatbevitel, 232, 234, 444, 451
konvertálás
fokokból radiánba, 246
karakterlánccá, 384
karakterláncot radiánba, 383
radiánból fokokba, 246
radián, 246, 328
szögletes zárójelek ([ ]), karakterek
szögletes zárójelek között, 523
szóközök
AutoLISP-ben, 196
forgatókönyv fájlokban, 159
szórt háttérvilágítás intenzitása, 535
szótárak
csoportkódok, 589
következő tétel megkeresése, 423
névobjektum, 476
objektumok, 300
szöveg
a menüfájlban, 85
a párbeszédpanelekben, 635, 645
ablakok, vezérlés, 231
beszúrási pont, 240
fájlok
AutoLISP alkalmazások tárolása, 161
parancsok olvasása, 158
fontok, 61
írás
állapotsorba, 458
képernyőmenü-területekre, 458
kiterjedés, 239
kiválasztási halmazok, 534
szöveges ablakok vezérlése, 231
szöveg alapvonala, 240
szöveg beillesztési pontja, 240
SZÖVEG parancs, 198
szürkeárnyalatú képek, importálás, 559
szűrők, kiválasztási halmaz, 264, 267, 501
szűrőlista argumentum (ssget függvény), 264
mód kiválasztási opciók, 267
szűrővizsgálatok,logikai csoportosítás, 270

T

T előre definiált változó, 200
és globális karakter összehasonlítás, 204
tab karakter (\t), 202
tábla adatok kezelése, 533
Tábla mód, 257
TÁBLA parancs, 124, 257
táblák, kalibrálás, 257
táblamenük, 124
TABMODE rendszerváltozó, 257
tabulálás, AutoLISP programfájlokban, 196
tabulátor karakter (\t), 203
tabulátorok AutoLISP programfájlokban, 196
tangens. *Lásd* AutoLISP függvények, árkusz tangens, 409
tárgyraszter mód, 435, 484, 499
távoli fényforrás, létrehozás, 537
távolság, a bővített adatban, 293
területek a párbeszédpanelekben, 643
teszt. *Lásd* ellenőrzés
tétel lezárása, 95
TEXT rajzelemtípus, csoportkódok, 585
TEXTEVAL rendszerváltozó, 198
TGA képek
megjelenítés, 547
renderelt képek elmentése, 546
TIFF fájlformátum, 561
TIFF képek
megjelenítés, 547
renderelt képek elmentése, 546
TIFFBE parancs, 561
a kép szélesség-magasság arányának megváltoztatása, 558
élvizsgálat, 559
használt színek száma, 559
háttérszínek meghatározása, 558
importálás
világosságon alapuló képek, 560
tiltott címkék, 115
több DCL fájl, betöltés, 469
több vektor, rajzolás, 459
TOLERANCE rajzelemtípus, csoportkódok, 586
törlés
ASE csatolás, 403
azonosítók, 404, 406
elérési útvonal, 397
fények, 535
jelenetek, 542
nézetablak, 454
objektumok, 427
kiválasztási halmazok, 498
szöveges képernyő, 514
törtjel. *Lásd* visszahajló törtjel (\)
TRACE rajzelemtípus, csoportkódok, 586
TÜKRÖZ3D parancs, 542

U

UCS rajzelemtípus, csoportkódok, 594
ÚJ parancs, 22
Új rajz létrehozása párbeszédpanel, 22
újsor, írás a parancssorba, 513
UNDOCTL rendszerváltozó, 217
Unicode fontok, 71
Unicode karaktersorozat \U+XXXX, 202
UNITMODE rendszerváltozó, 245
üzenetek

megjelenítés a parancssorban, 227
Lásd még hibaüzenetek
üzenetfájl, 36

V

Válasszon alakfájlt párbeszédpanel, 50
Válasszon egy LISP fájlt párbeszédpanel, 448
választható argumentumok, 215
valós számok, 192, 347
bővített adatban, 291
DIESEL, 153, 154
ellenőrzés, 482
felhasználói adat bevitele, 452
felhasználói adatbevitel, 233, 234
konverziós függvények, 243, 250, 409, 425, 442, 492
váltás
grafikusról szöveges képernyőre, 514
modelltér és papírtér között, 144
változók, 193
AutoLISP, 197
előre definiált, 198
és ! (felkiáltójel), 197
környezet, 226
megjelenítés, 332
nil értéke, 197
parancssorban, 197
rendszer, 226
változók megjelenítése, 332
VÉGE parancs, helyettesítés, 219
végrehajtási függvények (ASE), 261
vektor grafikus kép, megjelenítés párbeszédpanelben, 673
vektorok
grafikus képernyőre rajzolás, 454, 459
rajzolás a párbeszédpanel-ábrába, 519
verem, 563
VERTEX rajzelemtípus, csoportkódok, 587
verzió
ASE, 408
AutoLISP, 519
vesszővel elválasztott karakterláncok, 153
vezérlőkarakter (\e), 202
vezérlőkarakterek
DCL karakterláncokban megengedett, 650
karakterláncokban, 201
menütételekben, 99
vezérlőkarakterekláncok, bővített adatban, 292
VIEW rajzelemtípus, csoportkódok, 594
VIEWPORT rajzelemtípus
létrehozás, 431
megváltoztatás, 434
VIEWPORT rajzelemtípus, csoportkódok, 587
Világ koordináta-rendszer. *Lásd* VKR
világosság, képek importálása, 560
világtér
pozició, bővített adatban, 292
világtérbeli irány, a bővített adatban, 293
virtuális függvénylapozás, 520
VISLKONV parancs, 562
Vissza jellemző, 158
VISSZA parancs, 158
és hibák, 217
visszaállítás, objektumok, 427
visszahajló törtjel (\)
és a vezérlőkarakterek nyomtatása, 485
karakterláncokban, 192, 201, 202, 245, 311, 418, 453
mint PAUSE szimbólum, 198, 225, 418
visszajelzések, 307
indítékkódok, 307
kezelés csúszkákból, 312
Lásd még DCL zónák, előre definiált attribútumok, akció
Visual Link adatok, AutoCAD használatához történő konvertálás, 562
VKR, 251, 253, 275, 477, 515
irányok, bővített adatban, 292
pozició bővített adatban, 292
VLÁNC parancs, 213, 333
VLEDIT parancs, 279, 286
VONAL parancs, 198, 213
vonalak
Lásd még metszése
metszése, 466
pontozott, 38
szaggatott, 38, 45
vonalláncok
a körvonalak kitöltése PostScript kitöltési mintával, 549
definíciós adatok, 477
entget függvény és, 429
entmod függvény és, 434
görbe, 286
képernyő ábrájának frissítése, 436
kiválasztás, 477
létrehozás, 283, 431
megváltoztatás, 434
spline, 286
vonaltípusok
definíniós fájlok, 38
egyszerű, 38
összetett, 38
alakok, 41
szöveg, 42
VPORT rajzelemtípus, csoportkódok, 595
VPORT szimbólumtábla, 288, 299
VTIPUS parancs, 38

W

Windows
a "plusz vagy mínusz" szimbólum megjelenítése, 204
alkalmazások, indítás, 505
hivatkozás menükre, 127
menüfájlok, 127
menük vezérlése, 228
súgófájlok, 29, 134

X

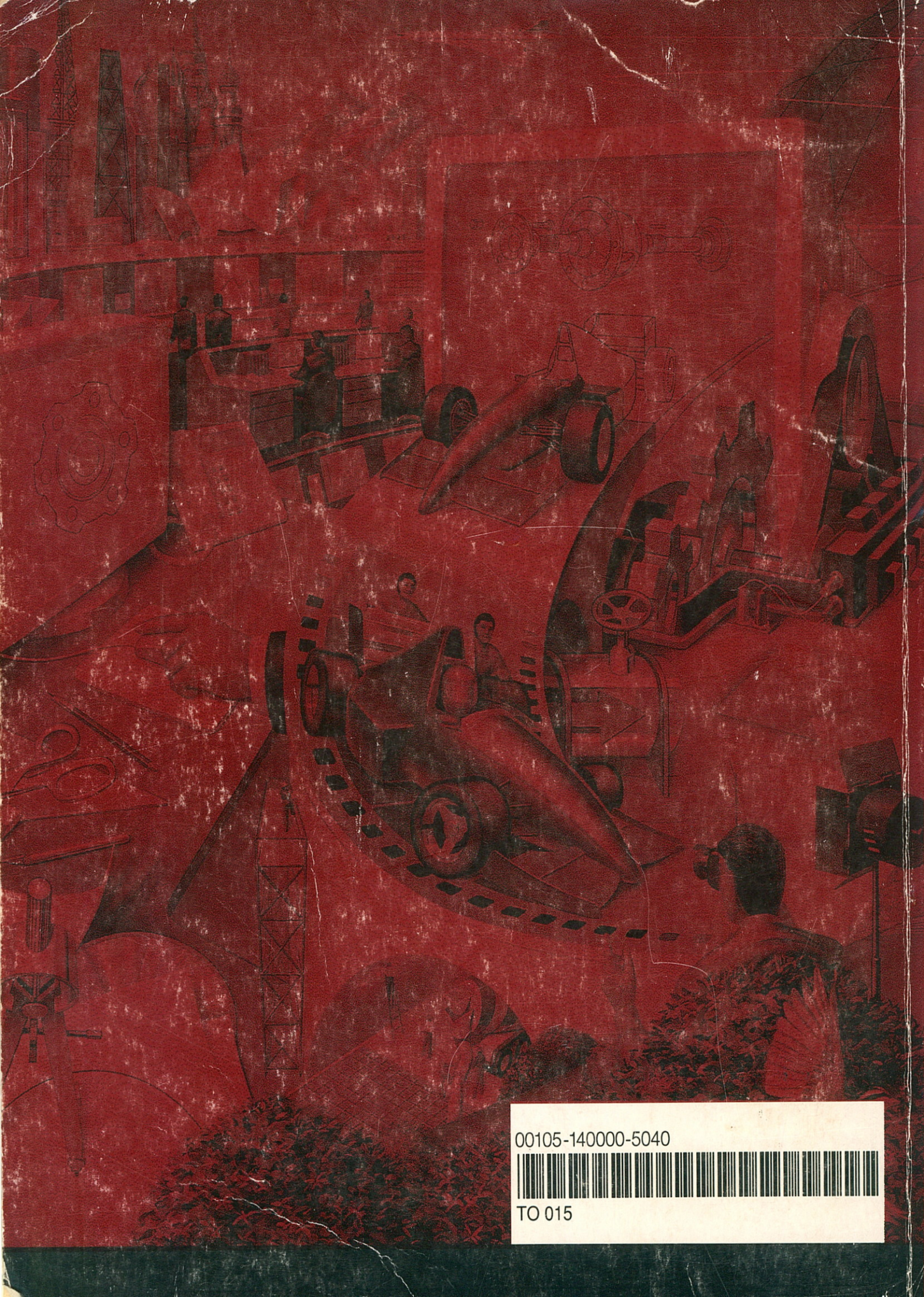
xdata
    Lásd még bővített adatok
XLINE rajzelemtípus, csoportkódok, 587

Y

Y koordináta, hozzájutás, 415, 467

Z

Z koordináta, hozzájutás, 415, 467
zárójelek
    AutoLISP-ben
        függvények, 330
    listákban, 192
zárójelek ([ ]), karakterek zárójelek között, 523
zónák, 309
    akciókifejezések, 305
    alkalmazás-specifikus adat, 323
    függvény áttekintés, 309
    letiltás, 310
    módok és értékek
        inicializálás, 309
        megváltoztatás visszajelzés során, 310
    visszajelzések, 309
    Lásd még DCL zónák
zónák (párbeszédpanel)
    függvény áttekintés, 369, 371
    gyakorló példa függvény, 340
    Lásd még ábrazónák
zónakifejezés változók, 305



00105-140000-5040

TO 015